Package 'tidylearn'

February 6, 2026

Title A Unified Tidy Interface to R's Machine Learning Ecosystem

Version 0.1.0

Description Provides a unified tidyverse-compatible interface to R's machine learning packages. Wraps established implementations from 'glmnet', 'randomForest', 'xgboost', 'e1071', 'rpart', 'gbm', 'nnet', 'cluster', 'dbscan', and others - providing consistent function signatures, tidy tibble output, and unified 'ggplot2'-based visualization. The underlying algorithms are unchanged; 'tidylearn' simply makes them easier to use together. Access raw model objects via the $fit slot for package-specific functionality. Methods include random forests Breiman (2001) <doi:10.1023/A:1010933404324>, LASSO regression Tibshirani (1996) <doi:10.1111/j.2517-6161.1996.tb02080.x>, elastic net Zou and Hastie (2005) <doi:10.1111/j.1467-9868.2005.00503.x>, support vector machines Cortes and Vapnik (1995) <doi:10.1007/BF00994018>, and gradient boosting Friedman (2001) <doi:10.1214/aos/1013203451>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 3.6.0)

Imports dplyr (>= 1.0.0), ggplot2 (>= 3.3.0), tibble (>= 3.0.0), tidyr (>= 1.0.0), purrr (>= 0.3.0), rlang (>= 0.4.0), magrittr, stats, e1071, gbm, glmnet, nnet, randomForest, rpart, rsample, ROCR, yardstick, cluster (>= 2.1.0), dbscan (>= 1.1.0), MASS, smacof (>= 2.1.0)

Suggests arules, arulesViz, car, caret, DT, GGally, ggforce, gridExtra, keras, knitr, lmtest, mclust, moments, NeuralNetTools, onnx, parsnip, recipes, reticulate, rmarkdown, rpart.plot, scales, shiny, shinydashboard, tensorflow, testthat (>= 3.0.0), workflows, xgboost

Config/testthat/edition 3

URL https://github.com/ces0491/tidylearn

BugReports https://github.com/ces0491/tidylearn/issues

VignetteBuilder knitr

Collate 'utils.R' 'core.R' 'preprocessing.R'
'supervised-classification.R' 'supervised-regression.R'
'supervised-regularization.R' 'supervised-trees.R'
'supervised-svm.R' 'supervised-neural-networks.R'
'supervised-deep-learning.R' 'supervised-xgboost.R'
'unsupervised-distance.R' 'unsupervised-pca.R'
'unsupervised-mds.R' 'unsupervised-clustering.R'
'unsupervised-hclust.R' 'unsupervised-dbscan.R'
'unsupervised-market-basket.R' 'unsupervised-validation.R'
'integration.R' 'pipeline.R' 'model-selection.R' 'tuning.R'
'interactions.R' 'diagnostics.R' 'metrics.R' 'visualization.R'
'workflows.R'

NeedsCompilation no

Author Cesaire Tobias [aut, cre]

Maintainer Cesaire Tobias <cesaire@sheetsolved.com>

Repository CRAN

Date/Publication 2026-02-06 13:50:02 UTC

Contents

`augment_dbscan` *Augment Data with DBSCAN Cluster Assignments*

Description

Augment Data with DBSCAN Cluster Assignments

Usage

```
augment_dbscan(dbscan_obj, data)
```

Arguments

dbscan_obj	A tidy_dbscan object
data	Original data frame

Value

Original data with cluster information added

---

augment_hclust *Augment Data with Hierarchical Cluster Assignments*

Description

Add cluster assignments to original data

Usage

```
augment_hclust(hclust_obj, data, k = NULL, h = NULL)
```

Arguments

hclust_obj	A tidy_hclust object
data	Original data frame
k	Number of clusters (optional)
h	Height at which to cut (optional)

Value

Original data with cluster column added

augment_kmeans *Augment Data with K-Means Cluster Assignments*

Description

Augment Data with K-Means Cluster Assignments

Usage

```
augment_kmeans(kmeans_obj, data)
```

Arguments

kmeans_obj	A tidy_kmeans object
data	Original data frame

Value

Original data with cluster column added

augment_pam *Augment Data with PAM Cluster Assignments*

Description

Augment Data with PAM Cluster Assignments

Usage

```
augment_pam(pam_obj, data)
```

Arguments

pam_obj	A tidy_pam object
data	Original data frame

Value

Original data with cluster column added

augment_pca *Augment Original Data with PCA Scores*

Description

Add PC scores to the original dataset

Usage

```
augment_pca(pca_obj, data, n_components = NULL)
```

Arguments

pca_obj	A tidy_pca object
data	Original data frame
n_components	Number of PCs to add (default: all)

Value

Original data with PC scores added

calc_validation_metrics *Calculate Cluster Validation Metrics*

Description

Comprehensive validation metrics for a clustering result

Usage

```
calc_validation_metrics(clusters, data = NULL, dist_mat = NULL)
```

Arguments

clusters	Vector of cluster assignments
data	Original data frame (for WSS calculation)
dist_mat	Distance matrix (for silhouette)

Value

A tibble with validation metrics

calc_wss *Calculate Within-Cluster Sum of Squares for Different k*

Description

Used for elbow method to determine optimal k

Usage

```
calc_wss(data, max_k = 10, nstart = 25)
```

Arguments

data	A data frame or tibble
max_k	Maximum number of clusters to test (default: 10)
nstart	Number of random starts for each k (default: 25)

Value

A tibble with k and corresponding total within-cluster SS

compare_clusterings *Compare Multiple Clustering Results*

Description

Compare Multiple Clustering Results

Usage

```
compare_clusterings(cluster_list, data, dist_mat = NULL)
```

Arguments

cluster_list	Named list of cluster assignment vectors
data	Original data
dist_mat	Distance matrix

Value

A tibble comparing all clustering results

compare_distances *Compare Distance Methods*

Description

Compute distances using multiple methods for comparison

Usage

```
compare_distances(data, methods = c("euclidean", "manhattan", "maximum"))
```

Arguments

data	A data frame or tibble
methods	Character vector of methods to compare

Value

A list of dist objects named by method

create_cluster_dashboard *Create Summary Dashboard*

Description

Generate a multi-panel summary of clustering results

Usage

```
create_cluster_dashboard(
  data,
  cluster_col = "cluster",
  validation_metrics = NULL
)
```

Arguments

data	Data frame with cluster assignments
cluster_col	Cluster column name
validation_metrics	Optional tibble of validation metrics

Value

Combined plot grid

explore_dbscan_params *Explore DBSCAN Parameters*

Description

Test multiple eps and minPts combinations

Usage

```
explore_dbscan_params(data, eps_values, minPts_values)
```

Arguments

data	A data frame or matrix
eps_values	Vector of eps values to test
minPts_values	Vector of minPts values to test

Value

A tibble with parameter combinations and resulting cluster counts

filter_rules_by_item *Filter Rules by Item*

Description

Subset rules containing specific items

Usage

```
filter_rules_by_item(rules_obj, item, where = "both")
```

Arguments

rules_obj	A tidy_apriori object or tibble of rules
item	Character; item to filter by
where	Character; "lhs", "rhs", or "both" (default: "both")

Value

A tibble of filtered rules

find_related_items *Find Related Items*

Description

Find items frequently purchased with a given item

Usage

```
find_related_items(rules_obj, item, min_lift = 1.5, top_n = 10)
```

Arguments

rules_obj	A tidy_apriori object
item	Character; item to find associations for
min_lift	Minimum lift threshold (default: 1.5)
top_n	Number of top associations to return (default: 10)

Value

A tibble of related items with association metrics

get_pca_loadings *Get PCA Loadings in Wide Format*

Description

Get PCA Loadings in Wide Format

Usage

```
get_pca_loadings(pca_obj, n_components = NULL)
```

Arguments

pca_obj	A tidy_pca object
n_components	Number of components to include (default: all)

Value

A tibble with loadings in wide format

get_pca_variance *Get Variance Explained Summary*

Description

Get Variance Explained Summary

Usage

```
get_pca_variance(pca_obj)
```

Arguments

pca_obj	A tidy_pca object

Value

A tibble with variance statistics

inspect_rules *Inspect Association Rules*

Description

View rules sorted by various quality measures

Usage

```
inspect_rules(rules_obj, by = "lift", n = 10, decreasing = TRUE)
```

Arguments

rules_obj	A tidy_apriori object or rules object
by	Sort by: "support", "confidence", "lift" (default), "count"
n	Number of rules to display (default: 10)
decreasing	Sort in decreasing order? (default: TRUE)

Value

A tibble of top rules

optimal_clusters *Find Optimal Number of Clusters*

Description

Use multiple methods to suggest optimal k

Usage

```
optimal_clusters(data, max_k = 10, methods = c("silhouette", "gap", "wss"))
```

Arguments

data	A data frame or tibble
max_k	Maximum k to test (default: 10)
methods	Vector of methods: "silhouette", "gap", "wss" (default: all)

Value

A list with results from each method

optimal_hclust_k *Determine Optimal Number of Clusters for Hierarchical Clustering*

Description

Use silhouette or gap statistic to find optimal k

Usage

```
optimal_hclust_k(hclust_obj, method = "silhouette", max_k = 10)
```

Arguments

hclust_obj	A tidy_hclust object
method	Character; "silhouette" (default) or "gap"
max_k	Maximum number of clusters to test (default: 10)

Value

A list with optimal k and evaluation results

plot.tidylearn_eda *Plot EDA results*

Description

Plot EDA results

Usage

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

Arguments

x	A tidylearn_eda object
...	Additional arguments (ignored)

Value

Invisibly returns the input object x, called for side effects (plotting)

plot.tidylearn_model *Plot method for tidylearn models*

Description

Plot method for tidylearn models

Usage

```
## S3 method for class 'tidylearn_model'
plot(x, type = "auto", ...)
```

Arguments

x	A tidylearn model object
type	Plot type (default: "auto")
...	Additional arguments passed to plotting functions

Value

A ggplot2 object or NULL, called primarily for side effects

plot_clusters *Plot Clusters in 2D Space*

Description

Visualize clustering results using first two dimensions or specified dimensions

Usage

```
plot_clusters(
  data,
  cluster_col = "cluster",
  x_col = NULL,
  y_col = NULL,
  centers = NULL,
  title = "Cluster Plot",
  color_noise_black = TRUE
)
```

Arguments

data	A data frame with cluster assignments
cluster_col	Name of cluster column (default: "cluster")
x_col	X-axis variable (if NULL, uses first numeric column)
y_col	Y-axis variable (if NULL, uses second numeric column)
centers	Optional data frame of cluster centers
title	Plot title
color_noise_black	If TRUE, color noise points (cluster 0) black

Value

A ggplot object

plot_cluster_comparison *Create Cluster Comparison Plot*

Description

Compare multiple clustering results side-by-side

Usage

```
plot_cluster_comparison(data, cluster_cols, x_col, y_col)
```

Arguments

data	Data frame with multiple cluster columns
cluster_cols	Vector of cluster column names
x_col	X-axis variable
y_col	Y-axis variable

Value

A grid of ggplot objects

---

plot_cluster_sizes *Plot Cluster Size Distribution*

Description

Create bar plot of cluster sizes

Usage

```
plot_cluster_sizes(clusters, title = "Cluster Size Distribution")
```

Arguments

clusters	Vector of cluster assignments
title	Plot title (default: "Cluster Size Distribution")

Value

A ggplot object

---

plot_dendrogram *Plot Dendrogram with Cluster Highlights*

Description

Enhanced dendrogram with colored cluster rectangles

Usage

```
plot_dendrogram(
  hclust_obj,
  k = NULL,
  title = "Hierarchical Clustering Dendrogram"
)
```

Arguments

hclust_obj	Hierarchical clustering object (hclust or tidy_hclust)
k	Number of clusters to highlight
title	Plot title

Value

Invisibly returns hclust object (plots as side effect)

---

plot_distance_heatmap *Create Distance Heatmap*

Description

Visualize distance matrix as heatmap

Usage

```
plot_distance_heatmap(
  dist_mat,
  cluster_order = NULL,
  title = "Distance Heatmap"
)
```

Arguments

dist_mat	Distance matrix (dist object)
cluster_order	Optional vector to reorder observations by cluster
title	Plot title

Value

A ggplot object

plot_elbow *Create Elbow Plot for K-Means*

Description

Plot total within-cluster sum of squares vs number of clusters

Usage

```
plot_elbow(wss_data, add_line = FALSE, suggested_k = NULL)
```

Arguments

wss_data	A tibble with columns k and tot_withinss (from calc_wss)
add_line	Add vertical line at suggested optimal k? (default: FALSE)
suggested_k	If add_line=TRUE, which k to highlight

Value

A ggplot object

plot_gap_stat *Plot Gap Statistic*

Description

Plot Gap Statistic

Usage

```
plot_gap_stat(gap_obj, show_methods = FALSE)
```

Arguments

gap_obj	A tidy_gap object
show_methods	Logical; show all three k selection methods? (default: FALSE)

Value

A ggplot object

plot_knn_dist *Plot k-NN Distance Plot*

Description

Visualize k-NN distances to help choose eps

Usage

```
plot_knn_dist(data, k = 4, add_suggestion = TRUE, percentile = 0.95)
```

Arguments

data	A data frame or tidy_knn_dist result
k	If data is a data frame, k for k-NN (default: 4)
add_suggestion	Add suggested eps line? (default: TRUE)
percentile	Percentile for suggestion (default: 0.95)

Value

A ggplot object

plot_mds *Plot MDS Configuration*

Description

Visualize MDS results

Usage

```
plot_mds(mds_obj, color_by = NULL, label_points = TRUE, dim_x = 1, dim_y = 2)
```

Arguments

mds_obj	A tidy_mds object
color_by	Optional variable to color points by
label_points	Logical; add point labels? (default: TRUE)
dim_x	Which dimension for x-axis (default: 1)
dim_y	Which dimension for y-axis (default: 2)

Value

A ggplot object

plot_silhouette *Plot Silhouette Analysis*

Description

Plot Silhouette Analysis

Usage

```
plot_silhouette(sil_obj)
```

Arguments

sil_obj	A tidy_silhouette object or tibble from tidy_silhouette_analysis

Value

A ggplot object

plot_variance_explained *Plot Variance Explained (PCA)*

Description

Create combined scree plot showing individual and cumulative variance

Usage

```
plot_variance_explained(variance_tbl, threshold = 0.8)
```

Arguments

variance_tbl	Variance tibble from tidy_pca
threshold	Horizontal line for variance threshold (default: 0.8 for 80%)

Value

A ggplot object

`predict.tidylearn_model` *Predict using a tidylearn model*

Description

Unified prediction interface for both supervised and unsupervised models

Usage

```
## S3 method for class 'tidylearn_model'
predict(object, new_data = NULL, type = "response", ...)
```

Arguments

`object`	A tidylearn model object
`new_data`	A data frame containing the new data. If NULL, uses training data.
`type`	Type of prediction. For supervised: "response" (default), "prob", "class". For unsupervised: "scores", "clusters", "transform" depending on method.
`...`	Additional arguments

Value

Predictions as a tibble

`predict.tidylearn_stratified` *Predict from stratified models*

Description

Predict from stratified models

Usage

```
## S3 method for class 'tidylearn_stratified'
predict(object, new_data = NULL, ...)
```

Arguments

`object`	A tidylearn_stratified model object
`new_data`	New data for predictions
`...`	Additional arguments

Value

A tibble of predictions with cluster assignments

predict.tidylearn_transfer *Predict with transfer learning model*

Description

Predict with transfer learning model

Usage

```
## S3 method for class 'tidylearn_transfer'
predict(object, new_data, ...)
```

Arguments

object	A tidylearn_transfer model object
new_data	New data for predictions
...	Additional arguments

Value

A tibble of predictions

print.tidylearn_automl *Print auto ML results*

Description

Print auto ML results

Usage

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

Arguments

x	A tidylearn_automl object
...	Additional arguments (ignored)

Value

Invisibly returns the input object x

print.tidylearn_eda *Print EDA results*

Description

Print EDA results

Usage

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

Arguments

x	A tidylearn_eda object
...	Additional arguments (ignored)

Value

Invisibly returns the input object x

print.tidylearn_model *Print method for tidylearn models*

Description

Print method for tidylearn models

Usage

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

Arguments

x	A tidylearn model object
...	Additional arguments (ignored)

Value

Invisibly returns the input object x

print.tidylearn_pipeline *Print a tidylearn pipeline*

Description

Print a tidylearn pipeline

Usage

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

Arguments

x	A tidylearn pipeline object
...	Additional arguments (not used)

Value

Invisibly returns the pipeline

print.tidy_apriori *Print Method for tidy_apriori*

Description

Print Method for tidy_apriori

Usage

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

Arguments

x	A tidy_apriori object
...	Additional arguments (ignored)

Value

Invisibly returns the input object x

print.tidy_dbscan *Print Method for tidy_dbscan*

Description

Print Method for tidy_dbscan

Usage

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

Arguments

x	A tidy_dbscan object
...	Additional arguments (ignored)

Value

Invisibly returns the input object x

print.tidy_gap *Print Method for tidy_gap*

Description

Print Method for tidy_gap

Usage

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

Arguments

x	A tidy_gap object
...	Additional arguments (ignored)

Value

Invisibly returns the input object x

print.tidy_hclust *Print Method for tidy_hclust*

Description

Print Method for tidy_hclust

Usage

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

Arguments

x	A tidy_hclust object
...	Additional arguments (ignored)

Value

Invisibly returns the input object x

print.tidy_kmeans *Print Method for tidy_kmeans*

Description

Print Method for tidy_kmeans

Usage

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

Arguments

x	A tidy_kmeans object
...	Additional arguments (ignored)

Value

Invisibly returns the input object x

print.tidy_mds *Print Method for tidy_mds*

Description

Print Method for tidy_mds

Usage

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

Arguments

x	A tidy_mds object
...	Additional arguments (ignored)

Value

Invisibly returns the input object x

print.tidy_pam *Print Method for tidy_pam*

Description

Print Method for tidy_pam

Usage

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

Arguments

x	A tidy_pam object
...	Additional arguments (ignored)

Value

Invisibly returns the input object x

print.tidy_pca *Print Method for tidy_pca*

Description

Print Method for tidy_pca

Usage

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

Arguments

x	A tidy_pca object
...	Additional arguments (ignored)

Value

Invisibly returns the input object x

print.tidy_silhouette *Print Method for tidy_silhouette*

Description

Print Method for tidy_silhouette

Usage

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

Arguments

x	A tidy_silhouette object
...	Additional arguments (ignored)

Value

Invisibly returns the input object x

recommend_products *Generate Product Recommendations*

Description

Get product recommendations based on basket contents

Usage

```
recommend_products(rules_obj, basket, top_n = 5, min_confidence = 0.5)
```

Arguments

rules_obj	A tidy_apriori object
basket	Character vector of items in current basket
top_n	Number of recommendations to return (default: 5)
min_confidence	Minimum confidence threshold (default: 0.5)

Value

A tibble with recommended items and metrics

standardize_data *Standardize Data*

Description

Center and/or scale numeric variables

Usage

```
standardize_data(data, center = TRUE, scale = TRUE)
```

Arguments

data	A data frame or tibble
center	Logical; center variables? (default: TRUE)
scale	Logical; scale variables to unit variance? (default: TRUE)

Value

A tibble with standardized numeric variables

suggest_eps *Suggest eps Parameter for DBSCAN*

Description

Use k-NN distance plot to suggest eps value

Usage

```
suggest_eps(data, minPts = 5, method = "percentile", percentile = 0.95)
```

Arguments

data	A data frame or matrix
minPts	Minimum points parameter (used as k for k-NN)
method	Method to suggest eps: "knee" (default), "percentile"
percentile	If method="percentile", which percentile to use (default: 0.95)

Value

A list containing:

- eps: suggested epsilon value
- knn_distances: full tibble of k-NN distances
- method: method used

Examples

```
eps_info <- suggest_eps(iris, minPts = 5)
eps_info$eps
```

summarize_rules *Summarize Association Rules*

Description

Get summary statistics about rules

Usage

```
summarize_rules(rules_obj)
```

Arguments

rules_obj	A tidy_apriori object or rules tibble

Value

A list with summary statistics

`summary.tidylearn_model` *Summary method for tidylearn models*

Description

Summary method for tidylearn models

Usage

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

Arguments

`object`	A tidylearn model object
`...`	Additional arguments (ignored)

Value

Invisibly returns the input object

`summary.tidylearn_pipeline` *Summarize a tidylearn pipeline*

Description

Summarize a tidylearn pipeline

Usage

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

Arguments

`object`	A tidylearn pipeline object
`...`	Additional arguments (not used)

Value

Invisibly returns the pipeline

`tidylearn-classification` *Classification Functions for tidylearn*

Description

Logistic regression and classification metrics functionality

`tidylearn-core` *tidylearn: A Unified Tidy Interface to R's Machine Learning Ecosystem*

Description

Core functionality for tidylearn. This package provides a unified tidyverse-compatible interface to established R machine learning packages including glmnet, randomForest, xgboost, e1071, rpart, gbm, nnet, cluster, and dbscan. The underlying algorithms are unchanged - tidylearn wraps them with consistent function signatures, tidy tibble output, and unified ggplot2-based visualization. Access raw model objects via model$fit.

`tidylearn-deep-learning` *Deep Learning for tidylearn*

Description

Deep learning functionality using Keras/TensorFlow

`tidylearn-diagnostics` *Advanced Diagnostics Functions for tidylearn*

Description

Functions for advanced model diagnostics, assumption checking, and outlier detection

`tidylearn-interactions` *Interaction Analysis Functions for tidylearn*

Description

Functions for testing, visualizing, and analyzing interactions

`tidylearn-metrics` *Metrics Functionality for tidylearn*

Description

Functions for calculating model evaluation metrics

`tidylearn-model-selection` *Model Selection Functions for tidylearn*

Description

Functions for stepwise model selection, cross-validation, and hyperparameter tuning

`tidylearn-neural-networks` *Neural Networks for tidylearn*

Description

Neural network functionality for classification and regression

`tidylearn-pipeline` *Model Pipeline Functions for tidylearn*

Description

Functions for creating end-to-end model pipelines

tidylearn-regression *Regression Functions for tidylearn*

Description

Linear and polynomial regression functionality

tidylearn-regularization *Regularization Functions for tidylearn*

Description

Ridge, Lasso, and Elastic Net regularization functionality

tidylearn-svm *Support Vector Machines for tidylearn*

Description

SVM functionality for classification and regression

tidylearn-trees *Tree-based Methods for tidylearn*

Description

Decision trees, random forests, and boosting functionality

tidylearn-tuning *Hyperparameter Tuning Functions for tidylearn*

Description

Functions for automatic hyperparameter tuning and selection

tidylearn-visualization *Visualization Functions for tidylearn*

Description

General visualization functions for tidylearn models

tidylearn-xgboost *XGBoost Functions for tidylearn*

Description

XGBoost-specific implementation for gradient boosting

tidy_apriori *Tidy Apriori Algorithm*

Description

Mine association rules using the Apriori algorithm with tidy output

Usage

```
tidy_apriori(
  transactions,
  support = 0.01,
  confidence = 0.5,
  minlen = 2,
  maxlen = 10,
  target = "rules"
)
```

Arguments

transactions	A transactions object or data frame
support	Minimum support (default: 0.01)
confidence	Minimum confidence (default: 0.5)
minlen	Minimum rule length (default: 2)
maxlen	Maximum rule length (default: 10)
target	Type of association mined: "rules" (default), "frequent itemsets", "maximally frequent itemsets"

Value

A list of class "tidy_rules" containing:

- rules_tbl: tibble of rules with lhs, rhs, and quality measures
- rules: original rules object
- parameters: parameters used

Examples

```
data("Groceries", package = "arules")

# Basic apriori
rules <- tidy_apriori(Groceries, support = 0.001, confidence = 0.5)

# Access rules
rules$rules_tbl
```

---

tidy_clara *Tidy CLARA (Clustering Large Applications)*

Description

Performs CLARA clustering (scalable version of PAM)

Usage

```
tidy_clara(data, k, metric = "euclidean", samples = 50, sampsize = NULL)
```

Arguments

data	A data frame or tibble
k	Number of clusters
metric	Distance metric (default: "euclidean")
samples	Number of samples to draw (default: 50)
sampsize	Sample size (default: min(n, 40 + 2*k))

Value

A list of class "tidy_clara" containing clustering results

Examples

```
# CLARA for large datasets
large_data <- iris[rep(1:nrow(iris), 10), 1:4]
clara_result <- tidy_clara(large_data, k = 3, samples = 50)
print(clara_result)
```

`tidy_cutree` *Cut Hierarchical Clustering Tree*

Description

Cut dendrogram to obtain cluster assignments

Usage

```
tidy_cutree(hclust_obj, k = NULL, h = NULL)
```

Arguments

`hclust_obj`	A tidy_hclust object or hclust object
`k`	Number of clusters (optional)
`h`	Height at which to cut (optional)

Value

A tibble with observation IDs and cluster assignments

`tidy_dbscan` *Tidy DBSCAN Clustering*

Description

Performs density-based clustering with tidy output

Usage

```
tidy_dbscan(data, eps, minPts = 5, cols = NULL, distance = "euclidean")
```

Arguments

`data`	A data frame, tibble, or distance matrix
`eps`	Neighborhood radius (epsilon)
`minPts`	Minimum number of points to form a dense region (default: 5)
`cols`	Columns to include (tidy select). If NULL, uses all numeric columns.
`distance`	Distance metric if data is not a dist object (default: "euclidean")

Value

A list of class "tidy_dbscan" containing:

- clusters: tibble with observation IDs and cluster assignments (0 = noise)
- core_points: logical vector indicating core points
- n_clusters: number of clusters (excluding noise)
- n_noise: number of noise points
- model: original dbscan object

Examples

```
# Basic DBSCAN
db_result <- tidy_dbscan(iris, eps = 0.5, minPts = 5)

# With suggested eps from k-NN distance plot
eps_suggestion <- suggest_eps(iris, minPts = 5)
db_result <- tidy_dbscan(iris, eps = eps_suggestion$eps, minPts = 5)
```

---

tidy_dendrogram *Plot Dendrogram*

Description

Create dendrogram visualization

Usage

```
tidy_dendrogram(hclust_obj, k = NULL, hang = 0.01, cex = 0.7)
```

Arguments

hclust_obj	A tidy_hclust object or hclust object
k	Optional; number of clusters to highlight with rectangles
hang	Fraction of plot height to hang labels (default: 0.01)
cex	Label size (default: 0.7)

Value

Invisibly returns the hclust object (plots as side effect)

tidy_dist *Tidy Distance Matrix Computation*

Description

Compute distance matrices with tidy output

Usage

```
tidy_dist(data, method = "euclidean", cols = NULL, ...)
```

Arguments

data	A data frame or tibble
method	Character; distance method (default: "euclidean"). Options: "euclidean", "manhattan", "maximum", "gower"
cols	Columns to include (tidy select). If NULL, uses all numeric columns.
...	Additional arguments passed to distance functions

Value

A dist object with tidy attributes

tidy_gap_stat *Tidy Gap Statistic*

Description

Compute gap statistic for determining optimal number of clusters

Usage

```
tidy_gap_stat(data, FUN_cluster = NULL, max_k = 10, B = 50, nstart = 25)
```

Arguments

data	A data frame or tibble
FUN_cluster	Clustering function (default: uses kmeans internally)
max_k	Maximum number of clusters (default: 10)
B	Number of bootstrap samples (default: 50)
nstart	If using kmeans, number of random starts (default: 25)

Value

A list of class "tidy_gap" containing gap statistics

`tidy_gower` *Gower Distance Calculation*

Description

Computes Gower distance for mixed data types (numeric, factor, ordered)

Usage

```
tidy_gower(data, weights = NULL)
```

Arguments

`data`	A data frame or tibble
`weights`	Optional named vector of variable weights (default: equal weights)

Details

Gower distance handles mixed data types:

- Numeric: range-normalized Manhattan distance
- Factor/Character: 0 if same, 1 if different
- Ordered: treated as numeric ranks

Formula: d_ij = sum(w_k * d_ijk) / sum(w_k) where d_ijk is the dissimilarity for variable k between obs i and j

Value

A dist object containing Gower distances

Examples

```
# Create example data with mixed types
car_data <- data.frame(
  horsepower = c(130, 250, 180),
  weight = c(1200, 1650, 1420),
  color = factor(c("red", "black", "blue"))
)

# Compute Gower distance
gower_dist <- tidy_gower(car_data)
```

`tidy_hclust` *Tidy Hierarchical Clustering*

Description

Performs hierarchical clustering with tidy output

Usage

```
tidy_hclust(data, method = "average", distance = "euclidean", cols = NULL)
```

Arguments

`data`	A data frame, tibble, or dist object
`method`	Agglomeration method: "ward.D2", "single", "complete", "average" (default), "mcquitty", "median", "centroid"
`distance`	Distance metric if data is not a dist object (default: "euclidean")
`cols`	Columns to include (tidy select). If NULL, uses all numeric columns.

Value

A list of class "tidy_hclust" containing:

- model: hclust object
- dist: distance matrix used
- method: linkage method used
- data: original data (for plotting)

Examples

```
# Basic hierarchical clustering
hc_result <- tidy_hclust(USArrests, method = "average")

# With specific distance
hc_result <- tidy_hclust(mtcars, method = "complete", distance = "manhattan")
```

`tidy_kmeans` *Tidy K-Means Clustering*

Description

Performs k-means clustering with tidy output

Usage

```
tidy_kmeans(
  data,
  k,
  cols = NULL,
  nstart = 25,
  iter_max = 100,
  algorithm = "Hartigan-Wong"
)
```

Arguments

`data`	A data frame or tibble
`k`	Number of clusters
`cols`	Columns to include (tidy select). If NULL, uses all numeric columns.
`nstart`	Number of random starts (default: 25)
`iter_max`	Maximum number of iterations (default: 100)
`algorithm`	K-means algorithm: "Hartigan-Wong" (default), "Lloyd", "Forgy", "MacQueen"

Value

A list of class "tidy_kmeans" containing:

- clusters: tibble with observation IDs and cluster assignments
- centers: tibble of cluster centers
- metrics: tibble with clustering quality metrics
- model: original kmeans object

Examples

```
# Basic k-means
km_result <- tidy_kmeans(iris, k = 3)
```

tidy_knn_dist *Compute k-NN Distances*

Description

Calculate distances to k-th nearest neighbor for each point

Usage

```
tidy_knn_dist(data, k = 4, cols = NULL)
```

Arguments

data	A data frame or matrix
k	Number of nearest neighbors (default: 4)
cols	Columns to include (tidy select). If NULL, uses all numeric columns.

Value

A tibble with observation IDs and k-NN distances

tidy_mds *Tidy Multidimensional Scaling*

Description

Unified interface for MDS methods with tidy output

Usage

```
tidy_mds(data, method = "classical", ndim = 2, distance = "euclidean", ...)
```

Arguments

data	A data frame, tibble, or distance matrix
method	Character; "classical" (default), "metric", "nonmetric", "sammon", or "kruskal"
ndim	Number of dimensions for output (default: 2)
distance	Character; distance metric if data is not already a dist object (default: "euclidean")
...	Additional arguments passed to specific MDS functions

Value

A list of class "tidy_mds" containing:

- config: tibble of MDS configuration (coordinates)
- stress: goodness-of-fit measure (if applicable)
- method: character string of method used
- model: original model object

Examples

```
# Classical MDS
mds_result <- tidy_mds(eurodist, method = "classical")
print(mds_result)
```

---

tidy_mds_classical *Classical (Metric) MDS*

Description

Performs classical multidimensional scaling using cmdscale()

Usage

```
tidy_mds_classical(dist_mat, ndim = 2, add_rownames = TRUE)
```

Arguments

dist_mat	A distance matrix (dist object)
ndim	Number of dimensions (default: 2)
add_rownames	Preserve row names from distance matrix (default: TRUE)

Value

A tidy_mds object

tidy_mds_kruskal *Kruskal's Non-metric MDS*

Description

Performs Kruskal's isoMDS

Usage

```
tidy_mds_kruskal(dist_mat, ndim = 2, ...)
```

Arguments

dist_mat	A distance matrix (dist object)
ndim	Number of dimensions (default: 2)
...	Additional arguments passed to MASS::isoMDS()

Value

A tidy_mds object

tidy_mds_sammon *Sammon Mapping*

Description

Performs Sammon's non-linear mapping

Usage

```
tidy_mds_sammon(dist_mat, ndim = 2, ...)
```

Arguments

dist_mat	A distance matrix (dist object)
ndim	Number of dimensions (default: 2)
...	Additional arguments passed to MASS::sammon()

Value

A tidy_mds object

tidy_mds_smacof *SMACOF MDS (Metric or Non-metric)*

Description

Performs MDS using SMACOF algorithm from the smacof package

Usage

```
tidy_mds_smacof(dist_mat, ndim = 2, type = "ratio", ...)
```

Arguments

dist_mat	A distance matrix (dist object)
ndim	Number of dimensions (default: 2)
type	Character; "ratio" for metric, "ordinal" for non-metric (default: "ratio")
...	Additional arguments passed to smacof::mds()

Value

A tidy_mds object

tidy_pam *Tidy PAM (Partitioning Around Medoids)*

Description

Performs PAM clustering with tidy output

Usage

```
tidy_pam(data, k, metric = "euclidean", cols = NULL)
```

Arguments

data	A data frame, tibble, or dist object
k	Number of clusters
metric	Distance metric (default: "euclidean"). Use "gower" for mixed data types.
cols	Columns to include (tidy select). If NULL, uses all columns.

Value

A list of class "tidy_pam" containing:

- clusters: tibble with observation IDs and cluster assignments
- medoids: tibble of medoid indices and values
- silhouette: average silhouette width
- model: original pam object

Examples

```
# PAM with Euclidean distance
pam_result <- tidy_pam(iris, k = 3)

# PAM with Gower distance for mixed data
pam_result <- tidy_pam(mtcars, k = 3, metric = "gower")
```

---

tidy_pca *Tidy Principal Component Analysis*

Description

Performs PCA on a dataset using tidyverse principles. Returns a tidy list containing scores, loadings, variance explained, and the original model.

Usage

```
tidy_pca(data, cols = NULL, scale = TRUE, center = TRUE, method = "prcomp")
```

Arguments

data	A data frame or tibble
cols	Columns to include in PCA (tidy select syntax). If NULL, uses all numeric columns.
scale	Logical; should variables be scaled to unit variance? Default TRUE.
center	Logical; should variables be centered? Default TRUE.
method	Character; "prcomp" (default, recommended) or "princomp"

Value

A list of class "tidy_pca" containing:

- scores: tibble of PC scores with observation identifiers
- loadings: tibble of variable loadings in long format
- variance: tibble of variance explained by each PC
- model: the original prcomp/princomp object
- settings: list of scale, center, method used

Examples

```
# Basic PCA
pca_result <- tidy_pca(USArrests)


# Access components
pca_result$scores
pca_result$loadings
pca_result$variance
```

`tidy_pca_biplot` *Create PCA Biplot*

Description

Visualize both observations and variables in PC space

Usage

```
tidy_pca_biplot(
  pca_obj,
  pc_x = 1,
  pc_y = 2,
  color_by = NULL,
  arrow_scale = 1,
  label_obs = FALSE,
  label_vars = TRUE
)
```

Arguments

`pca_obj`	A tidy_pca object
`pc_x`	Principal component for x-axis (default: 1)
`pc_y`	Principal component for y-axis (default: 2)
`color_by`	Optional column name to color points by
`arrow_scale`	Scaling factor for variable arrows (default: 1)
`label_obs`	Logical; label observations? (default: FALSE)
`label_vars`	Logical; label variables? (default: TRUE)

Value

A ggplot object

tidy_pca_screeplot *Create PCA Scree Plot*

Description

Visualize variance explained by each principal component

Usage

```
tidy_pca_screeplot(pca_obj, type = "proportion", add_line = TRUE)
```

Arguments

pca_obj	A tidy_pca object
type	Character; "variance" or "proportion" (default)
add_line	Logical; add horizontal line at eigenvalue = 1? (for Kaiser criterion)

Value

A ggplot object

tidy_rules *Convert Association Rules to Tidy Tibble*

Description

Convert Association Rules to Tidy Tibble

Usage

```
tidy_rules(rules)
```

Arguments

rules	A rules object from arules

Value

A tibble with one row per rule

tidy_silhouette *Tidy Silhouette Analysis*

Description

Compute silhouette statistics for cluster validation

Usage

```
tidy_silhouette(clusters, dist_mat)
```

Arguments

clusters	Vector of cluster assignments
dist_mat	Distance matrix (dist object)

Value

A list of class "tidy_silhouette" containing:

- silhouette_data: tibble with silhouette values for each observation
- avg_width: average silhouette width
- cluster_avg: average silhouette width by cluster

tidy_silhouette_analysis *Silhouette Analysis Across Multiple k Values*

Description

Silhouette Analysis Across Multiple k Values

Usage

```
tidy_silhouette_analysis(
  data,
  max_k = 10,
  method = "kmeans",
  nstart = 25,
  dist_method = "euclidean",
  linkage_method = "average"
)
```

Arguments

data	A data frame or tibble
max_k	Maximum number of clusters to test (default: 10)
method	Clustering method: "kmeans" (default) or "hclust"
nstart	If kmeans, number of random starts (default: 25)
dist_method	Distance metric (default: "euclidean")
linkage_method	If hclust, linkage method (default: "average")

Value

A tibble with k and average silhouette widths

---

tl_add_cluster_features *Cluster-Based Features*

Description

Add cluster assignments as features for supervised learning. This semi-supervised approach can capture non-linear patterns.

Usage

```
tl_add_cluster_features(data, response = NULL, method = "kmeans", ...)
```

Arguments

data	A data frame
response	Response variable name (will be excluded from clustering)
method	Clustering method: "kmeans", "pam", "hclust", "dbscan"
...	Additional arguments for clustering

Value

Original data with cluster assignment column(s) added

Examples

```
# Add cluster features before supervised learning
data_with_clusters <- tl_add_cluster_features(iris, response = "Species",
                                              method = "kmeans", k = 3)
model <- tl_model(data_with_clusters, Species ~ ., method = "forest")
```

`tl_anomaly_aware` *Anomaly-Aware Supervised Learning*

Description

Detect outliers using DBSCAN or other methods, then optionally remove them or down-weight them before supervised learning.

Usage

```
tl_anomaly_aware(
  data,
  formula,
  response,
  anomaly_method = "dbscan",
  action = "flag",
  supervised_method = "logistic",
  ...
)
```

Arguments

`data`	A data frame
`formula`	Model formula
`response`	Response variable name
`anomaly_method`	Method for anomaly detection: "dbscan", "isolation_forest"
`action`	Action to take: "remove", "flag", "downweight"
`supervised_method`	Supervised learning method
`...`	Additional arguments

Value

A tidylearn model or list with model and anomaly info

Examples

```
model <- tl_anomaly_aware(iris, Species ~ ., response = "Species",
                          anomaly_method = "dbscan", action = "flag")
```

tl_auto_interactions *Find important interactions automatically*

Description

Find important interactions automatically

Usage

```
tl_auto_interactions(
  data,
  formula,
  top_n = 3,
  min_r2_change = 0.01,
  max_p_value = 0.05,
  exclude_vars = NULL
)
```

Arguments

data	A data frame containing the data
formula	A formula specifying the base model without interactions
top_n	Number of top interactions to return
min_r2_change	Minimum change in R-squared to consider
max_p_value	Maximum p-value for significance
exclude_vars	Character vector of variables to exclude from interaction testing

Value

A tidylearn model with important interactions

tl_auto_ml *High-Level Workflows for Common Machine Learning Patterns*

Description

These functions provide end-to-end workflows that showcase tidylearn's ability to seamlessly combine multiple learning paradigms Auto ML: Automated Machine Learning Workflow

Usage

```
tl_auto_ml(
  data,
  formula,
  task = "auto",
  use_reduction = TRUE,
  use_clustering = TRUE,
  time_budget = 300,
  cv_folds = 5,
  metric = NULL
)
```

Arguments

data	A data frame
formula	Model formula (for supervised learning)
task	Task type: "classification", "regression", or "auto" (default)
use_reduction	Whether to try dimensionality reduction (default: TRUE)
use_clustering	Whether to add cluster features (default: TRUE)
time_budget	Time budget in seconds (default: 300)
cv_folds	Number of cross-validation folds (default: 5)
metric	Evaluation metric (default: auto-selected based on task)

Details

Automatically explores multiple modeling approaches including dimensionality reduction, clustering, and various supervised methods. Returns the best performing model based on cross-validation.

Value

Best model with performance comparison

Examples

```
# Automated modeling
result <- tl_auto_ml(iris, Species ~ .)
best_model <- result$best_model
result$leaderboard
```

tl_calc_classification_metrics *Calculate classification metrics*

Description

Calculate classification metrics

Usage

```
tl_calc_classification_metrics(
  actuals,
  predicted,
  predicted_probs = NULL,
  metrics = c("accuracy", "precision", "recall", "f1", "auc"),
  thresholds = NULL,
  ...
)
```

Arguments

actuals	Actual values (ground truth)
predicted	Predicted class values
predicted_probs	Predicted probabilities (for metrics like AUC)
metrics	Character vector of metrics to compute
thresholds	Optional vector of thresholds to evaluate for threshold-dependent metrics
...	Additional arguments

Value

A tibble of evaluation metrics

tl_check_assumptions *Check model assumptions*

Description

Check model assumptions

Usage

```
tl_check_assumptions(model, test = TRUE, verbose = TRUE)
```

Arguments

model	A tidylearn model object
test	Logical; whether to perform statistical tests
verbose	Logical; whether to print test results and explanations

Value

A list with assumption check results

---

tl_compare_cv *Compare models using cross-validation*

Description

Compare models using cross-validation

Usage

```
tl_compare_cv(data, models, folds = 5, metrics = NULL, ...)
```

Arguments

data	A data frame containing the training data
models	A list of tidylearn model objects
folds	Number of cross-validation folds
metrics	Character vector of metrics to compute
...	Additional arguments

Value

A tibble with cross-validation results for all models

tl_compare_pipeline_models *Compare models from a pipeline*

Description

Compare models from a pipeline

Usage

```
tl_compare_pipeline_models(pipeline, metrics = NULL)
```

Arguments

pipeline	A tidylearn pipeline object with results
metrics	Character vector of metrics to compare (if NULL, uses all available)

Value

A comparison plot of model performance

tl_cv *Cross-validation for tidylearn models*

Description

Cross-validation for tidylearn models

Usage

```
tl_cv(data, formula, method, folds = 5, ...)
```

Arguments

data	Data frame
formula	Model formula
method	Modeling method
folds	Number of cross-validation folds
...	Additional arguments

Value

Cross-validation results

tl_dashboard *Create interactive visualization dashboard for a model*

Description

Create interactive visualization dashboard for a model

Usage

```
tl_dashboard(model, new_data = NULL, ...)
```

Arguments

model	A tidylearn model object
new_data	Optional data frame for evaluation (if NULL, uses training data)
...	Additional arguments

Value

A Shiny app object

tl_default_param_grid *Create pre-defined parameter grids for common models*

Description

Create pre-defined parameter grids for common models

Usage

```
tl_default_param_grid(method, size = "medium", is_classification = TRUE)
```

Arguments

method	Model method ("tree", "forest", "boost", "svm", etc.)
size	Grid size: "small", "medium", "large"
is_classification	Whether the task is classification or regression

Value

A named list of parameter values to tune

tl_detect_outliers *Detect outliers in the data*

Description

Detect outliers in the data

Usage

```
tl_detect_outliers(
  data,
  variables = NULL,
  method = "iqr",
  threshold = NULL,
  plot = TRUE
)
```

Arguments

data	A data frame containing the data
variables	Character vector of variables to check for outliers
method	Method for outlier detection: "boxplot", "z-score", "cook", "iqr", "mahalanobis"
threshold	Threshold for outlier detection
plot	Logical; whether to create a plot of outliers

Value

A list with outlier detection results

tl_diagnostic_dashboard *Create a comprehensive diagnostic dashboard*

Description

Create a comprehensive diagnostic dashboard

Usage

```
tl_diagnostic_dashboard(
  model,
  include_influence = TRUE,
  include_assumptions = TRUE,
  include_performance = TRUE,
  arrange_plots = "grid"
)
```

Arguments

`model`	A tidylearn model object
`include_influence`	Logical; whether to include influence diagnostics
`include_assumptions`	Logical; whether to include assumption checks
`include_performance`	Logical; whether to include performance metrics
`arrange_plots`	Layout arrangement (e.g., "grid", "row", "column")

Value

A plot grid with diagnostic plots

`tl_evaluate` *Evaluate a tidylearn model*

Description

Evaluate a tidylearn model

Usage

```
tl_evaluate(object, new_data = NULL, ...)
```

Arguments

`object`	A tidylearn model object
`new_data`	Optional new data for evaluation (if NULL, uses training data)
`...`	Additional arguments

Value

A tibble of evaluation metrics

tl_explore *Exploratory Data Analysis Workflow*

Description

Comprehensive EDA combining unsupervised learning techniques to understand data structure before modeling

Usage

```
tl_explore(data, response = NULL, max_components = 5, k_range = 2:6)
```

Arguments

data	A data frame
response	Optional response variable for colored visualizations
max_components	Maximum PCA components to compute (default: 5)
k_range	Range of k values for clustering (default: 2:6)

Value

An EDA object with multiple analyses

Examples

```
eda <- tl_explore(iris, response = "Species")
plot(eda)
```

tl_get_best_model *Get the best model from a pipeline*

Description

Get the best model from a pipeline

Usage

```
tl_get_best_model(pipeline)
```

Arguments

pipeline	A tidylearn pipeline object with results

Value

The best tidylearn model

tl_influence_measures *Calculate influence measures for a linear model*

Description

Calculate influence measures for a linear model

Usage

```
tl_influence_measures(
  model,
  threshold_cook = NULL,
  threshold_leverage = NULL,
  threshold_dffits = NULL
)
```

Arguments

model	A tidylearn model object
threshold_cook	Cook's distance threshold (default: 4/n)
threshold_leverage	Leverage threshold (default: 2*(p+1)/n)
threshold_dffits	DFFITS threshold (default: 2*sqrt((p+1)/n))

Value

A data frame with influence measures

tl_interaction_effects *Calculate partial effects based on a model with interactions*

Description

Calculate partial effects based on a model with interactions

Usage

```
tl_interaction_effects(model, var, by_var, at_values = NULL, intervals = TRUE)
```

Arguments

model	A tidylearn model object
var	Variable to calculate effects for
by_var	Variable to calculate effects by (interaction variable)
at_values	Named list of values at which to hold other variables
intervals	Logical; whether to include confidence intervals

Value

A data frame with marginal effects

---

tl_load_pipeline *Load a pipeline from disk*

Description

Load a pipeline from disk

Usage

```
tl_load_pipeline(file)
```

Arguments

file	Path to the pipeline file

Value

A tidylearn pipeline object

---

tl_model *Create a tidylearn model*

Description

Unified interface for creating machine learning models by wrapping established R packages. This function dispatches to the appropriate underlying package based on the method specified.

Usage

```
tl_model(data, formula = NULL, method = "linear", ...)
```

Arguments

data	A data frame containing the training data
formula	A formula specifying the model. For unsupervised methods, use ~ vars or NULL.
method	The modeling method. Supervised: "linear" (stats::lm), "logistic" (stats::glm), "tree" (rpart), "forest" (randomForest), "boost" (gbm), "ridge"/"lasso"/"elastic_net" (glmnet), "svm" (e1071), "nn" (nnet), "deep" (keras), "xgboost" (xgboost). Unsupervised: "pca" (stats::prcomp), "mds" (stats/MASS/smacof), "kmeans" (stats::kmeans), "pam"/"clara" (cluster), "hclust" (stats::hclust), "dbscan" (dbscan).
...	Additional arguments passed to the underlying model function

Details

The wrapped packages include: stats (lm, glm, prcomp, kmeans, hclust), glmnet, randomForest, xgboost, gbm, e1071, nnet, rpart, cluster, and dbscan. The underlying algorithms are unchanged - this function provides a consistent interface and returns tidy output.

Access the raw model object from the underlying package via model$fit.

Value

A tidylearn model object containing the fitted model ($fit), specification, and training data

Examples

```
# Classification -> wraps randomForest::randomForest()
model <- tl_model(iris, Species ~ ., method = "forest")
model$fit  # Access the raw randomForest object

# Regression -> wraps stats::lm()
model <- tl_model(mtcars, mpg ~ wt + hp, method = "linear")
model$fit  # Access the raw lm object

# PCA -> wraps stats::prcomp()
model <- tl_model(iris, ~ ., method = "pca")
model$fit  # Access the raw prcomp object

# Clustering -> wraps stats::kmeans()
model <- tl_model(iris, method = "kmeans", k = 3)
model$fit  # Access the raw kmeans object
```

---

tl_pipeline *Create a modeling pipeline*

Description

Create a modeling pipeline

Usage

```
tl_pipeline(
  data,
  formula,
  preprocessing = NULL,
  models = NULL,
  evaluation = NULL,
  ...
)
```

Arguments

data	A data frame containing the data
formula	A formula specifying the model
preprocessing	A list of preprocessing steps
models	A list of models to train
evaluation	A list of evaluation criteria
...	Additional arguments

Value

A tidylearn pipeline object

---

tl_plot_cv_comparison *Plot comparison of cross-validation results*

Description

Plot comparison of cross-validation results

Usage

```
tl_plot_cv_comparison(cv_results, metrics = NULL)
```

Arguments

cv_results	Results from tl_compare_cv function
metrics	Character vector of metrics to plot (if NULL, plots all metrics)

Value

A ggplot object

tl_plot_cv_results *Plot cross-validation results*

Description

Plot cross-validation results

Usage

```
tl_plot_cv_results(cv_results, metrics = NULL)
```

Arguments

cv_results	Cross-validation results from tl_cv function
metrics	Character vector of metrics to plot (if NULL, plots all metrics)

Value

A ggplot object with cross-validation results

tl_plot_deep_architecture *Plot deep learning model architecture*

Description

Plot deep learning model architecture

Usage

```
tl_plot_deep_architecture(model, ...)
```

Arguments

model	A tidylearn deep learning model object
...	Additional arguments

Value

A plot of the deep learning model architecture

tl_plot_deep_history *Plot deep learning model training history*

Description

Plot deep learning model training history

Usage

```
tl_plot_deep_history(model, metrics = c("loss", "val_loss"), ...)
```

Arguments

model	A tidylearn deep learning model object
metrics	Which metrics to plot (default: c("loss", "val_loss"))
...	Additional arguments

Value

A ggplot object with training history

tl_plot_gain *Plot gain chart for a classification model*

Description

Plot gain chart for a classification model

Usage

```
tl_plot_gain(model, new_data = NULL, bins = 10, ...)
```

Arguments

model	A tidylearn classification model object
new_data	Optional data frame for evaluation (if NULL, uses training data)
bins	Number of bins for grouping predictions (default: 10)
...	Additional arguments

Value

A ggplot object with gain chart

tl_plot_importance_comparison

Plot feature importance across multiple models

Description

Plot feature importance across multiple models

Usage

```
tl_plot_importance_comparison(..., top_n = 10, names = NULL)
```

Arguments

...	tidylearn model objects to compare
top_n	Number of top features to display (default: 10)
names	Optional character vector of model names

Value

A ggplot object with feature importance comparison

tl_plot_importance_regularized

Plot variable importance for a regularized regression model

Description

Plot variable importance for a regularized regression model

Usage

```
tl_plot_importance_regularized(model, lambda = "1se", top_n = 20, ...)
```

Arguments

model	A tidylearn regularized model object
lambda	Which lambda to use ("1se" or "min", default: "1se")
top_n	Number of top features to display (default: 20)
...	Additional arguments

Value

A ggplot object

tl_plot_influence *Plot influence diagnostics*

Description

Plot influence diagnostics

Usage

```
tl_plot_influence(
  model,
  plot_type = "cook",
  threshold_cook = NULL,
  threshold_leverage = NULL,
  threshold_dffits = NULL,
  n_labels = 3,
  label_size = 3
)
```

Arguments

model	A tidylearn model object
plot_type	Type of influence plot: "cook", "leverage", "index"
threshold_cook	Cook's distance threshold (default: 4/n)
threshold_leverage	Leverage threshold (default: 2*(p+1)/n)
threshold_dffits	DFFITS threshold (default: 2*sqrt((p+1)/n))
n_labels	Number of points to label (default: 3)
label_size	Text size for labels (default: 3)

Value

A ggplot object

tl_plot_interaction *Plot interaction effects*

Description

Plot interaction effects

Usage

```
tl_plot_interaction(
  model,
  var1,
  var2,
  n_points = 100,
  fixed_values = NULL,
  confidence = TRUE,
  ...
)
```

Arguments

model	A tidylearn model object
var1	First variable in the interaction
var2	Second variable in the interaction
n_points	Number of points to use for continuous variables
fixed_values	Named list of values for other variables in the model
confidence	Logical; whether to show confidence intervals
...	Additional arguments to pass to predict()

Value

A ggplot object

---

tl_plot_intervals *Create confidence and prediction interval plots*

Description

Create confidence and prediction interval plots

Usage

```
tl_plot_intervals(model, new_data = NULL, level = 0.95, ...)
```

Arguments

model	A tidylearn regression model object
new_data	Optional data frame for prediction (if NULL, uses training data)
level	Confidence level (default: 0.95)
...	Additional arguments

Value

A ggplot object

tl_plot_lift *Plot lift chart for a classification model*

Description

Plot lift chart for a classification model

Usage

```
tl_plot_lift(model, new_data = NULL, bins = 10, ...)
```

Arguments

model	A tidylearn classification model object
new_data	Optional data frame for evaluation (if NULL, uses training data)
bins	Number of bins for grouping predictions (default: 10)
...	Additional arguments

Value

A ggplot object with lift chart

tl_plot_model_comparison *Plot model comparison*

Description

Plot model comparison

Usage

```
tl_plot_model_comparison(..., new_data = NULL, metrics = NULL, names = NULL)
```

Arguments

...	tidylearn model objects to compare
new_data	Optional data frame for evaluation (if NULL, uses training data)
metrics	Character vector of metrics to compute
names	Optional character vector of model names

Value

A ggplot object with model comparison

tl_plot_nn_architecture *Plot neural network architecture*

Description

Plot neural network architecture

Usage

```
tl_plot_nn_architecture(model, ...)
```

Arguments

model	A tidylearn neural network model object
...	Additional arguments

Value

A ggplot object with neural network architecture

tl_plot_nn_tuning *Plot neural network training history*

Description

Plot neural network training history

Usage

```
tl_plot_nn_tuning(model, ...)
```

Arguments

model	A tidylearn neural network model object
...	Additional arguments

Value

A ggplot object with training history

tl_plot_partial_dependence

Plot partial dependence for tree-based models

Description

Plot partial dependence for tree-based models

Usage

```
tl_plot_partial_dependence(model, var, n.pts = 20, ...)
```

Arguments

model	A tidylearn tree-based model object
var	Variable name to plot
n.pts	Number of points for continuous variables (default: 20)
...	Additional arguments

Value

A ggplot object

tl_plot_regularization_cv

Plot cross-validation results for a regularized regression model

Description

Shows the cross-validation error as a function of lambda for ridge, lasso, or elastic net models fitted with cv.glmnet.

Usage

```
tl_plot_regularization_cv(model, ...)
```

Arguments

model	A tidylearn regularized model object (ridge, lasso, or elastic_net)
...	Additional arguments (currently unused)

Value

A ggplot object showing CV error vs lambda

tl_plot_regularization_path *Plot regularization path for a regularized regression model*

Description

Plot regularization path for a regularized regression model

Usage

```
tl_plot_regularization_path(model, label_n = 5, ...)
```

Arguments

model	A tidylearn regularized model object
label_n	Number of top features to label (default: 5)
...	Additional arguments

Value

A ggplot object

tl_plot_svm_boundary *Plot SVM decision boundary*

Description

Plot SVM decision boundary

Usage

```
tl_plot_svm_boundary(model, x_var = NULL, y_var = NULL, grid_size = 100, ...)
```

Arguments

model	A tidylearn SVM model object
x_var	Name of the x-axis variable
y_var	Name of the y-axis variable
grid_size	Number of points in each dimension for the grid (default: 100)
...	Additional arguments

Value

A ggplot object with decision boundary

tl_plot_svm_tuning *Plot SVM tuning results*

Description

Plot SVM tuning results

Usage

```
tl_plot_svm_tuning(model, ...)
```

Arguments

model	A tidylearn SVM model object
...	Additional arguments

Value

A ggplot object with tuning results

tl_plot_tree *Plot a decision tree*

Description

Plot a decision tree

Usage

```
tl_plot_tree(model, ...)
```

Arguments

model	A tidylearn tree model object
...	Additional arguments to pass to rpart.plot()

Value

A plot of the decision tree

tl_plot_tuning_results

Plot hyperparameter tuning results

Description

Plot hyperparameter tuning results

Usage

```
tl_plot_tuning_results(
  model,
  top_n = 5,
  param1 = NULL,
  param2 = NULL,
  plot_type = "scatter"
)
```

Arguments

model	A tidylearn model object with tuning results
top_n	Number of top parameter sets to highlight
param1	First parameter to plot (for 2D grid or scatter plots)
param2	Second parameter to plot (for 2D grid or scatter plots)
plot_type	Type of plot: "scatter", "grid", "parallel", "importance"

Value

A ggplot object

tl_plot_xgboost_importance

Plot feature importance for an XGBoost model

Description

Plot feature importance for an XGBoost model

Usage

```
tl_plot_xgboost_importance(model, top_n = 10, importance_type = "gain", ...)
```

Arguments

model	A tidylearn XGBoost model object
top_n	Number of top features to display (default: 10)
importance_type	Type of importance: "gain", "cover", "frequency"
...	Additional arguments

Value

A ggplot object

---

tl_plot_xgboost_shap_dependence *Plot SHAP dependence for a specific feature*

Description

Plot SHAP dependence for a specific feature

Usage

```
tl_plot_xgboost_shap_dependence(
  model,
  feature,
  interaction_feature = NULL,
  data = NULL,
  n_samples = 100
)
```

Arguments

model	A tidylearn XGBoost model object
feature	Feature name to plot
interaction_feature	Feature to use for coloring (default: NULL)
data	Data for SHAP value calculation (default: NULL, uses training data)
n_samples	Number of samples to use (default: 100, NULL for all)

Value

A ggplot object with SHAP dependence plot

tl_plot_xgboost_shap_summary

Plot SHAP summary for XGBoost model

Description

Plot SHAP summary for XGBoost model

Usage

```
tl_plot_xgboost_shap_summary(model, data = NULL, top_n = 10, n_samples = 100)
```

Arguments

model	A tidylearn XGBoost model object
data	Data for SHAP value calculation (default: NULL, uses training data)
top_n	Number of top features to display (default: 10)
n_samples	Number of samples to use (default: 100, NULL for all)

Value

A ggplot object with SHAP summary

tl_plot_xgboost_tree *Plot XGBoost tree visualization*

Description

Plot XGBoost tree visualization

Usage

```
tl_plot_xgboost_tree(model, tree_index = 0, ...)
```

Arguments

model	A tidylearn XGBoost model object
tree_index	Index of the tree to plot (default: 0, first tree)
...	Additional arguments

Value

Tree visualization

tl_predict_pipeline *Make predictions using a pipeline*

Description

Make predictions using a pipeline

Usage

```
tl_predict_pipeline(
  pipeline,
  new_data,
  type = "response",
  model_name = NULL,
  ...
)
```

Arguments

pipeline	A tidylearn pipeline object with results
new_data	A data frame containing the new data
type	Type of prediction (default: "response")
model_name	Name of model to use (if NULL, uses the best model)
...	Additional arguments passed to predict

Value

Predictions

tl_prepare_data *Data Preprocessing for tidylearn*

Description

Unified preprocessing functions that work with both supervised and unsupervised workflows Prepare Data for Machine Learning

Usage

```
tl_prepare_data(
  data,
  formula = NULL,
  impute_method = "mean",
  scale_method = "standardize",
  encode_categorical = TRUE,
  remove_zero_variance = TRUE,
  remove_correlated = FALSE,
  correlation_cutoff = 0.95
)
```

Arguments

data	A data frame
formula	Optional formula (for supervised learning)
impute_method	Method for missing value imputation: "mean", "median", "mode", "knn"
scale_method	Scaling method: "standardize", "normalize", "robust", "none"
encode_categorical	Whether to encode categorical variables (default: TRUE)
remove_zero_variance	Remove zero-variance features (default: TRUE)
remove_correlated	Remove highly correlated features (default: FALSE)
correlation_cutoff	Correlation threshold for removal (default: 0.95)

Details

Comprehensive preprocessing pipeline including imputation, scaling, encoding, and feature engineering

Value

A list containing processed data and preprocessing metadata

Examples

```
processed <- tl_prepare_data(iris, Species ~ ., scale_method = "standardize")
model <- tl_model(processed$data, Species ~ ., method = "logistic")
```

`tl_reduce_dimensions` *Integration Functions: Combining Supervised and Unsupervised Learning*

Description

These functions demonstrate the power of tidylearn's unified approach by seamlessly integrating supervised and unsupervised learning techniques. Feature Engineering via Dimensionality Reduction

Usage

```
tl_reduce_dimensions(
  data,
  response = NULL,
  method = "pca",
  n_components = NULL,
  ...
)
```

Arguments

`data`	A data frame
`response`	Response variable name (will be preserved)
`method`	Dimensionality reduction method: "pca", "mds"
`n_components`	Number of components to retain
`...`	Additional arguments for the dimensionality reduction method

Details

Use PCA, MDS, or other dimensionality reduction as a preprocessing step for supervised learning. This can improve model performance and interpretability.

Value

A list containing the transformed data and the reduction model

Examples

```
# Reduce dimensions before classification
reduced <- tl_reduce_dimensions(iris, response = "Species", method = "pca", n_components = 3)
model <- tl_model(reduced$data, Species ~ ., method = "logistic")
```

tl_run_pipeline *Run a tidylearn pipeline*

Description

Run a tidylearn pipeline

Usage

```
tl_run_pipeline(pipeline, verbose = TRUE)
```

Arguments

pipeline	A tidylearn pipeline object
verbose	Logical; whether to print progress

Value

A tidylearn pipeline with results

tl_save_pipeline *Save a pipeline to disk*

Description

Save a pipeline to disk

Usage

```
tl_save_pipeline(pipeline, file)
```

Arguments

pipeline	A tidylearn pipeline object
file	Path to save the pipeline

Value

Invisible NULL

`tl_semisupervised` *Semi-Supervised Learning via Clustering*

Description

Train a supervised model with limited labels by first clustering the data and propagating labels within clusters.

Usage

```
tl_semisupervised(
  data,
  formula,
  labeled_indices,
  cluster_method = "kmeans",
  supervised_method = "logistic",
  ...
)
```

Arguments

`data`	A data frame
`formula`	Model formula
`labeled_indices`	Indices of labeled observations
`cluster_method`	Clustering method for label propagation
`supervised_method`	Supervised learning method for final model
`...`	Additional arguments

Value

A tidylearn model trained on pseudo-labeled data

Examples

```
# Use only 10% of labels
labeled_idx <- sample(nrow(iris), size = 15)
model <- tl_semisupervised(iris, Species ~ ., labeled_indices = labeled_idx,
                           cluster_method = "kmeans", supervised_method = "logistic")
```

tl_split *Split data into train and test sets*

Description

Split data into train and test sets

Usage

```
tl_split(data, prop = 0.8, stratify = NULL, seed = NULL)
```

Arguments

data	A data frame
prop	Proportion for training set (default: 0.8)
stratify	Column name for stratified splitting
seed	Random seed for reproducibility

Value

A list with train and test data frames

Examples

```
split_data <- tl_split(iris, prop = 0.7, stratify = "Species")
train <- split_data$train
test <- split_data$test
```

tl_step_selection *Perform stepwise selection on a linear model*

Description

Perform stepwise selection on a linear model

Usage

```
tl_step_selection(
  data,
  formula,
  direction = "backward",
  criterion = "AIC",
  trace = FALSE,
  steps = 1000,
  ...
)
```

Arguments

data	A data frame containing the training data
formula	A formula specifying the initial model
direction	Direction of stepwise selection: "forward", "backward", or "both"
criterion	Criterion for selection: "AIC" or "BIC"
trace	Logical; whether to print progress
steps	Maximum number of steps to take
...	Additional arguments to pass to step()

Value

A selected model

---

tl_stratified_models *Stratified Features via Clustering*

Description

Create cluster-specific supervised models for heterogeneous data

Usage

```
tl_stratified_models(
  data,
  formula,
  cluster_method = "kmeans",
  k = 3,
  supervised_method = "linear",
  ...
)
```

Arguments

data	A data frame
formula	Model formula
cluster_method	Clustering method
k	Number of clusters
supervised_method	Supervised learning method
...	Additional arguments

Value

A list of models (one per cluster) plus cluster assignments

Examples

```
models <- tl_stratified_models(mtcars, mpg ~ ., cluster_method = "kmeans",
                               k = 3, supervised_method = "linear")
```

`tl_test_interactions` *Test for significant interactions between variables*

Description

Test for significant interactions between variables

Usage

```
tl_test_interactions(
  data,
  formula,
  var1 = NULL,
  var2 = NULL,
  all_pairs = FALSE,
  categorical_only = FALSE,
  numeric_only = FALSE,
  mixed_only = FALSE,
  alpha = 0.05
)
```

Arguments

`data`	A data frame containing the data
`formula`	A formula specifying the base model without interactions
`var1`	First variable to test for interactions
`var2`	Second variable to test for interactions (if NULL, tests var1 with all others)
`all_pairs`	Logical; whether to test all variable pairs
`categorical_only`	Logical; whether to only test categorical variables
`numeric_only`	Logical; whether to only test numeric variables
`mixed_only`	Logical; whether to only test numeric-categorical pairs
`alpha`	Significance level for interaction tests

Value

A data frame with interaction test results

tl_test_model_difference *Perform statistical comparison of models using cross-validation*

Description

Perform statistical comparison of models using cross-validation

Usage

```
tl_test_model_difference(
  cv_results,
  baseline_model = NULL,
  test = "t.test",
  metric = NULL
)
```

Arguments

cv_results	Results from tl_compare_cv function
baseline_model	Name of the model to use as baseline for comparison
test	Type of statistical test: "t.test" or "wilcox"
metric	Name of the metric to compare

Value

A data frame with statistical test results

tl_transfer_learning *Transfer Learning Workflow*

Description

Use unsupervised pre-training (e.g., autoencoder features) before supervised learning

Usage

```
tl_transfer_learning(
  data,
  formula,
  pretrain_method = "pca",
  supervised_method = "logistic",
  ...
)
```

Arguments

data	Training data
formula	Model formula
pretrain_method	Pre-training method: "pca", "autoencoder"
supervised_method	Supervised learning method
...	Additional arguments

Value

A transfer learning model

Examples

```
model <- tl_transfer_learning(iris, Species ~ ., pretrain_method = "pca")
```

---

tl_tune_deep *Tune a deep learning model*

Description

Tune a deep learning model

Usage

```
tl_tune_deep(
  data,
  formula,
  is_classification = FALSE,
  hidden_layers_options = list(c(32), c(64, 32), c(128, 64, 32)),
  learning_rates = c(0.01, 0.001, 1e-04),
  batch_sizes = c(16, 32, 64),
  epochs = 30,
  validation_split = 0.2,
  ...
)
```

Arguments

data	A data frame containing the training data
formula	A formula specifying the model
is_classification	Logical indicating if this is a classification problem

hidden_layers_options	List of vectors defining hidden layer configurations to try
learning_rates	Learning rates to try (default: c(0.01, 0.001, 0.0001))
batch_sizes	Batch sizes to try (default: c(16, 32, 64))
epochs	Number of training epochs (default: 30)
validation_split	Proportion of data for validation (default: 0.2)
...	Additional arguments

Value

A list with the best model and tuning results

`tl_tune_grid` *Tune hyperparameters for a model using grid search*

Description

Tune hyperparameters for a model using grid search

Usage

```
tl_tune_grid(
  data,
  formula,
  method,
  param_grid,
  folds = 5,
  metric = NULL,
  maximize = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

data	A data frame containing the training data
formula	A formula specifying the model
method	The modeling method to tune
param_grid	A named list of parameter values to tune
folds	Number of cross-validation folds
metric	Metric to optimize
maximize	Logical; whether to maximize (TRUE) or minimize (FALSE) the metric
verbose	Logical; whether to print progress
...	Additional arguments passed to tl_model

Value

A list with the best model and tuning results

`tl_tune_nn` *Tune a neural network model*

Description

Tune a neural network model

Usage

```
tl_tune_nn(
  data,
  formula,
  is_classification = FALSE,
  sizes = c(1, 2, 5, 10),
  decays = c(0, 0.001, 0.01, 0.1),
  folds = 5,
  ...
)
```

Arguments

`data`	A data frame containing the training data
`formula`	A formula specifying the model
`is_classification`	Logical indicating if this is a classification problem
`sizes`	Vector of hidden layer sizes to try
`decays`	Vector of weight decay parameters to try
`folds`	Number of cross-validation folds (default: 5)
`...`	Additional arguments to pass to nnet()

Value

A list with the best model and tuning results

`tl_tune_random` *Tune hyperparameters for a model using random search*

Description

Tune hyperparameters for a model using random search

Usage

```
tl_tune_random(
  data,
  formula,
  method,
  param_space,
  n_iter = 10,
  folds = 5,
  metric = NULL,
  maximize = NULL,
  verbose = TRUE,
  seed = NULL,
  ...
)
```

Arguments

`data`	A data frame containing the training data
`formula`	A formula specifying the model
`method`	The modeling method to tune
`param_space`	A named list of parameter spaces to sample from
`n_iter`	Number of random parameter combinations to try
`folds`	Number of cross-validation folds
`metric`	Metric to optimize
`maximize`	Logical; whether to maximize (TRUE) or minimize (FALSE) the metric
`verbose`	Logical; whether to print progress
`seed`	Random seed for reproducibility
`...`	Additional arguments passed to tl_model

Value

A list with the best model and tuning results

`tl_tune_xgboost` *Tune XGBoost hyperparameters*

Description

Tune XGBoost hyperparameters

Usage

```
tl_tune_xgboost(
  data,
  formula,
  is_classification = FALSE,
  param_grid = NULL,
  cv_folds = 5,
  early_stopping_rounds = 10,
  verbose = TRUE,
  ...
)
```

Arguments

`data`	A data frame containing the training data
`formula`	A formula specifying the model
`is_classification`	Logical indicating if this is a classification problem
`param_grid`	Named list of parameter values to try
`cv_folds`	Number of cross-validation folds (default: 5)
`early_stopping_rounds`	Early stopping rounds (default: 10)
`verbose`	Logical indicating whether to print progress (default: TRUE)
`...`	Additional arguments

Value

A list with the best model and tuning results

tl_version *Get tidylearn version information*

Description

Get tidylearn version information

Usage

```
tl_version()
```

Value

A package_version object containing the version number

tl_xgboost_shap *Generate SHAP values for XGBoost model interpretation*

Description

Generate SHAP values for XGBoost model interpretation

Usage

```
tl_xgboost_shap(model, data = NULL, n_samples = 100, trees_idx = NULL)
```

Arguments

model	A tidylearn XGBoost model object
data	Data for SHAP value calculation (default: NULL, uses training data)
n_samples	Number of samples to use (default: 100, NULL for all)
trees_idx	Trees to include (default: NULL, uses all trees)

Value

A data frame with SHAP values

`visualize_rules` *Visualize Association Rules*

Description

Create visualizations of association rules

Usage

```
visualize_rules(rules_obj, method = "scatter", top_n = 50, ...)
```

Arguments

`rules_obj`	A tidy_apriori object, rules object, or rules tibble
`method`	Visualization method: "scatter" (default), "graph", "grouped", "paracoord"
`top_n`	Number of top rules to visualize (default: 50)
`...`	Additional arguments passed to plot() for rules visualization

Value

Visualization (side effect) or ggplot object

Index