Package 'tidylearn'

May 8, 2026

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

Version 0.3.0

Description Provides a unified tidyverse-compatible interface to R's machine learning ecosystem - from data ingestion to model publishing. The tl_read() family reads data from files ('CSV', 'Excel', 'Parquet', 'JSON'), databases ('SQLite', 'PostgreSQL', 'MySQL', 'BigQuery'), and cloud sources ('S3', 'GitHub', 'Kaggle'). The tl_model() function wraps established implementations from 'glmnet', 'randomForest', 'xgboost', 'e1071', 'rpart', 'gbm', 'nnet', 'cluster', 'dbscan', and others with consistent function signatures and tidy tibble output. Results flow into unified 'ggplot2'-based visualization and optional formatted 'gt' tables via the tl_table() family. 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 7.3.3

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, bigrquery, car, caret, DBI, DT, GGally, ggforce, gridExtra, gt, jsonlite, keras, knitr, lmtest, mclust, moments, nanoparquet, NeuralNetTools, onnx, parsnip, paws.storage, readr, readxl, recipes, reticulate, RMariaDB, rmarkdown, RPostgres, rpart.plot, RSQLite, 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' 'read.R' 'read-backends.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' 'tables.R' 'workflows.R'

NeedsCompilation no

Author Cesaire Tobias [aut, cre]

Maintainer Cesaire Tobias <cesaire@sheetsolved.com>

Repository CRAN

Date/Publication 2026-04-09 09:30:02 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

A tibble containing the original `data` with additional columns `cluster` (factor), `is_noise` (logical), and `is_core` (logical).

Examples

```
db <- tidy_dbscan(iris[, 1:4], eps = 0.5, minPts = 5)
augmented <- augment_dbscan(db, iris)
```

`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

A tibble containing the original `data` with an additional `cluster` integer column indicating cluster assignments.

Examples

```
hc <- tidy_hclust(USArrests, method = "ward.D2")
augmented <- augment_hclust(hc, USArrests, k = 3)
```

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

A tibble containing the original `data` with an additional `cluster` factor column indicating cluster assignments.

Examples

```
km <- tidy_kmeans(iris[, 1:4], k = 3)
augmented <- augment_kmeans(km, iris)
```

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

A tibble containing the original `data` with an additional `cluster` factor column indicating cluster assignments.

Examples

```
pm <- tidy_pam(iris[, 1:4], k = 3)
augmented <- augment_pam(pm, iris)
```

---

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

A tibble containing the original `data` with additional columns for each principal component score (named `PC1`, `PC2`, etc.).

Examples

```
pca <- tidy_pca(USArrests)
augmented <- augment_pca(pca, USArrests, n_components = 2)
```

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 single-row tibble with columns k, min_size, max_size, avg_size, and optionally avg_silhouette, min_silhouette (if dist_mat provided), and total_wss (if data provided).

Examples

```
km <- kmeans(iris[, 1:4], centers = 3, nstart = 25)
d <- dist(iris[, 1:4])
metrics <- calc_validation_metrics(km$cluster, iris[, 1:4], d)
```

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 columns `k` (number of clusters) and `tot_withinss` (total within-cluster sum of squares).

Examples

```
wss <- calc_wss(iris[, 1:4], max_k = 6)
plot(wss$k, wss$tot_withinss, type = "b")
```

`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 with one row per clustering method and columns for each validation metric (see `calc_validation_metrics`), plus a `method` column identifying the clustering.

Examples

```
km3 <- kmeans(iris[, 1:4], 3, nstart = 25)$cluster
km4 <- kmeans(iris[, 1:4], 4, nstart = 25)$cluster
compare_clusterings(list(k3 = km3, k4 = km4), iris[, 1:4])
```

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 named list of `dist` objects, one per method.

Examples

```
dists <- compare_distances(iris[, 1:4], methods = c("euclidean", "manhattan"))
```

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

Invisibly returns a list of `ggplot` objects. The combined plot grid is drawn as a side effect via `grid.arrange`.

Examples

```
df <- iris[, 1:4]
df$cluster <- kmeans(df, 3)$cluster
create_cluster_dashboard(df)
```

`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 columns `eps`, `minPts`, `n_clusters`, `n_noise`, and `prop_noise` for each parameter combination.

Examples

```
params <- explore_dbscan_params(iris[, 1:4],
  eps_values = c(0.3, 0.5, 0.8), minPts_values = c(3, 5))
```

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 rules containing the specified `item` in the requested position.

Examples

```
if (requireNamespace("arules", quietly = TRUE)) {
  data("Groceries", package = "arules")
  res <- tidy_apriori(Groceries, support = 0.001, confidence = 0.5)
  filter_rules_by_item(res, "whole milk", where = "rhs")
}
```

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 rules involving the specified `item`, filtered by `min_lift` and sorted by lift in descending order.

Examples

```
if (requireNamespace("arules", quietly = TRUE)) {
  data("Groceries", package = "arules")
  res <- tidy_apriori(Groceries, support = 0.001, confidence = 0.5)
  find_related_items(res, "whole milk", min_lift = 1.5)
}
```

---

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 one row per variable and one column per principal component, containing the loading values.

Examples

```
pca <- tidy_pca(USArrests)
get_pca_loadings(pca, n_components = 2)
```

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 columns component, sdev, variance, prop_variance, and cum_variance.

Examples

```
pca <- tidy_pca(USArrests)
get_pca_variance(pca)
```

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 the top n rules sorted by the specified quality measure.

Examples

```
if (requireNamespace("arules", quietly = TRUE)) {
  data("Groceries", package = "arules")
  res <- tidy_apriori(Groceries, support = 0.001, confidence = 0.5)
  inspect_rules(res, by = "lift", n = 5)
}
```

---

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 of class "optimal_k_results" containing one or more of:

- wss: tibble from calc_wss (if "wss" method used)
- silhouette: tibble from tidy_silhouette_analysis (if "silhouette" method used)
- gap: a tidy_gap object from tidy_gap_stat (if "gap" method used)

Examples

```
opt <- optimal_clusters(iris[, 1:4], max_k = 6, methods = "wss")
```

`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 containing:

- optimal_k: the recommended number of clusters
- method: the evaluation method used
- values: numeric vector of evaluation scores (for silhouette)
- k_range: integer vector of k values tested (for silhouette)

If `method = "gap"`, returns a `tidy_gap` object instead.

Examples

```
hc <- tidy_hclust(USArrests, method = "ward.D2")
opt <- optimal_hclust_k(hc, method = "silhouette", max_k = 6)
```

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

The input object x, returned invisibly. Called for its side effect of plotting a PCA scatter plot coloured by cluster.

Examples

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

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 ggplot object. The specific plot depends on the model paradigm and type argument.

Examples

```
model <- tl_model(mtcars, mpg ~ wt + hp, method = "linear")
plot(model, type = "actual_predicted")
```

---

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.

Examples

```
km <- tidy_kmeans(iris[, 1:4], k = 3)
clustered <- augment_kmeans(km, iris[, 1:4])
plot_clusters(clustered)
```

`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

The return value of `grid.arrange`, a `gtable` drawn as a side effect.

Examples

```
df <- iris[, 1:4]
df$km3 <- kmeans(df, 3)$cluster
df$km4 <- kmeans(df, 4)$cluster
plot_cluster_comparison(df, c("km3", "km4"), "Sepal.Length", "Sepal.Width")
```

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.

Examples

```
clusters <- kmeans(iris[, 1:4], 3)$cluster
plot_cluster_sizes(clusters)
```

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 the hclust object. The dendrogram is drawn as a side effect.

Examples

```
hc <- hclust(dist(iris[, 1:4]))
plot_dendrogram(hc, k = 3)
```

`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.

Examples

```
d <- dist(iris[1:20, 1:4])
plot_distance_heatmap(d)
```

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.

Examples

```
wss <- data.frame(k = 2:6, tot_withinss = c(150, 90, 60, 50, 45))
plot_elbow(wss)
```

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.

Examples

```
gap <- tidy_gap_stat(iris[, 1:4], max_k = 6, B = 10)
plot_gap_stat(gap)
```

---

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.

Examples

```
plot_knn_dist(iris[, 1:4], k = 5)
```

---

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.

Examples

```
mds <- tidy_mds(USArrests, method = "classical")
plot_mds(mds)
```

---

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.

Examples

```
km <- kmeans(iris[, 1:4], centers = 3, nstart = 25)
d <- dist(iris[, 1:4])
sil <- tidy_silhouette(km$cluster, d)
plot_silhouette(sil)
```

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.

Examples

```
model <- tl_model(iris[, 1:4], method = "pca")
plot_variance_explained(model$fit$variance_explained)
```

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

A tibble with a `.pred` column containing predictions. For classification with `type = "prob"`, returns columns for each class probability.

Examples

```
model <- tl_model(mtcars, mpg ~ wt + hp, method = "linear")
predict(model)
predict(model, new_data = mtcars[1:5, ])
```

`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 with a `.pred` column containing predictions and a `.cluster` column with cluster assignments.

Examples

```
models <- tl_stratified_models(mtcars, mpg ~ .,
  cluster_method = "kmeans", k = 2, supervised_method = "linear")
preds <- predict(models)
```

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 with a .pred column containing predictions.

Examples

```
model <- tl_transfer_learning(iris, Species ~ .,
  pretrain_method = "pca", supervised_method = "logistic")
preds <- predict(model, iris[1:5, ])
```

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

The input object x, returned invisibly.

`print.tidylearn_data` *Print a tidylearn_data object*

Description

Print a tidylearn_data object

Usage

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

Arguments

x	A tidylearn_data object.
...	Additional arguments passed to the tibble print method.

Value

The input object x, returned invisibly.

Examples

```
f <- tempfile(fileext = ".csv")
write.csv(iris, f, row.names = FALSE)
d <- tl_read(f)
print(d)
```

`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

The input object x, returned invisibly.

Examples

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

---

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

The input object x, returned invisibly.

Examples

```
model <- tl_model(mtcars, mpg ~ wt + hp, method = "linear")
print(model)
```

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

The input pipeline object x, returned invisibly.

Examples

```
pipe <- tl_pipeline(iris, Species ~ .)
print(pipe)
```

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

The input object x, returned invisibly.

Examples

```
if (requireNamespace("arules", quietly = TRUE)) {
  data("Groceries", package = "arules")
  res <- tidy_apriori(Groceries, support = 0.001, confidence = 0.5)
  print(res)
}
```

---

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

The input object x, returned invisibly.

Examples

```
db <- tidy_dbscan(iris[, 1:4], eps = 0.5, minPts = 5)
print(db)
```

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

The input object x, returned invisibly.

Examples

```
gap <- tidy_gap_stat(iris[, 1:4], max_k = 6, B = 10)
print(gap)
```

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

The input object x, returned invisibly.

Examples

```
hc <- tidy_hclust(USArrests, method = "ward.D2")
print(hc)
```

---

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

The input object x, returned invisibly.

Examples

```
km <- tidy_kmeans(iris[, 1:4], k = 3)
print(km)
```

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

The input object x, returned invisibly.

Examples

```
mds <- tidy_mds(USArrests, method = "classical")
print(mds)
```

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

The input object x, returned invisibly.

Examples

```
pm <- tidy_pam(iris[, 1:4], k = 3)
print(pm)
```

`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

The input object x, returned invisibly.

Examples

```
pca <- tidy_pca(USArrests)
print(pca)
```

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

The input object x, returned invisibly.

Examples

```
km <- kmeans(iris[, 1:4], centers = 3, nstart = 25)
d <- dist(iris[, 1:4])
sil <- tidy_silhouette(km$cluster, d)
print(sil)
```

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 columns `rhs` (recommended item), `confidence`, `lift`, and `support`, sorted by lift in descending order.

Examples

```
if (requireNamespace("arules", quietly = TRUE)) {
  data("Groceries", package = "arules")
  res <- tidy_apriori(Groceries, support = 0.001, confidence = 0.5)
  recommend_products(res, basket = c("whole milk", "butter"))
}
```

---

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 numeric variables centered and/or scaled as specified; non-numeric columns are returned unchanged.

Examples

```
std <- standardize_data(iris[, 1:4])
```

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 `n_rules` and summary statistics (`min`, `max`, `mean`, `median`) for `support`, `confidence`, and `lift`.

Examples

```
if (requireNamespace("arules", quietly = TRUE)) {
  data("Groceries", package = "arules")
  res <- tidy_apriori(Groceries, support = 0.001, confidence = 0.5)
  summarize_rules(res)
}
```

---

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

The input `object`, returned invisibly. Called for its side effect of printing model summary and training performance.

Examples

```
model <- tl_model(mtcars, mpg ~ wt + hp, method = "linear")
summary(model)
```

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

The input pipeline `object`, returned invisibly. Called for its side effect of printing detailed pipeline and model results.

Examples

```
pipe <- tl_pipeline(iris, Species ~ .)
summary(pipe)
```

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-read *Data Reading Functions for tidylearn*

Description

Functions for reading data from diverse sources into tidy `tidylearn_data` objects. The main dispatcher `tl_read()` auto-detects the format from the file extension and routes to the appropriate reader. All readers return a `tidylearn_data` object, which is a tibble subclass carrying metadata about the data source.

Details

Supported file formats:

- **CSV**: `.csv` files via **readr** (with base R fallback)
- **TSV**: `.tsv` files via **readr** (with base R fallback)
- **Excel**: `.xls`, `.xlsx`, `.xlsm` files via **readxl**
- **Parquet**: `.parquet` files via **nanoparquet**
- **JSON**: `.json` files via **jsonlite**
- **RDS**: `.rds` files via base `readRDS()`
- **RData**: `.rdata`, `.rda` files via base `load()`

Supported databases (via **DBI**):

- **SQLite**: `.sqlite`, `.db` files via **RSQLite**
- **PostgreSQL**: via **RPostgres**
- **MySQL/MariaDB**: via **RMariaDB**
- **BigQuery**: via **bigrquery**

Supported cloud/API sources:

- **S3**: `s3://` URIs via **paws.storage**
- **GitHub**: raw file download from repositories
- **Kaggle**: dataset download via Kaggle CLI

Multi-file reading:

- **Multiple paths**: pass a character vector to `tl_read()`
- **Directories**: `tl_read_dir()` scans for data files with optional pattern/format filtering and recursive scanning
- **Zip archives**: `tl_read_zip()` extracts and reads from `.zip` files

When combining multiple files, a `source_file` column is added to identify the origin of each row.

`tidylearn-read-backends`

Data Reading Backends for tidylearn

Description

Backend readers for databases and cloud/API sources. All backends are optional dependencies checked at call time via `tl_check_packages()`.

Details

Database backends (via **DBI**):

- **SQLite**: via **RSQLite**
- **PostgreSQL**: via **RPostgres**
- **MySQL/MariaDB**: via **RMariaDB**
- **BigQuery**: via **bigrquery**

Cloud/API backends:

- **S3**: via **paws.storage**
- **GitHub**: via base `download.file()`
- **Kaggle**: via Kaggle CLI

---

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-tables *Table Functions for tidylearn*

Description

Functions for producing formatted gt tables from tidylearn models. Provides a parallel interface to the plot functions: `tl_table(model, type)` dispatches to the appropriate table formatter based on model type. Requires the gt package (suggested dependency).

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

Description

Functions providing end-to-end workflows that showcase tidylearn's ability to seamlessly combine multiple learning paradigms

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

```
if (requireNamespace("arules", quietly = TRUE)) {
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:

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

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 columns `.obs_id` (observation identifier) and `cluster` (integer cluster assignment).

Examples

```
hc <- tidy_hclust(USArrests, method = "ward.D2")
clusters <- tidy_cutree(hc, k = 3)
```

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

The hclust object, returned invisibly. The dendrogram is plotted as a side effect.

Examples

```
hc <- tidy_hclust(USArrests, method = "ward.D2")
tidy_dendrogram(hc, k = 3)
```

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 containing the computed distance matrix.

Examples

```
d <- tidy_dist(iris[, 1:4], method = "euclidean")
```

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_data: tibble with gap statistics for each k
- k_firstSEmax: optimal k via firstSEmax method (most conservative)
- k_globalmax: optimal k via globalmax method
- k_firstmax: optimal k via firstmax method
- recommended_k: recommended k (uses firstSEmax)
- model: the clusGap result

Examples

```
gap <- tidy_gap_stat(iris[, 1:4], max_k = 6, B = 10)
gap$recommended_k
```

---

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, with the method attribute set to "gower".

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 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 columns `.obs_id` (observation identifier), `knn_dist` (distance to k-th nearest neighbor), and `rank` (rank of the k-NN distance).

Examples

```
knn <- tidy_knn_dist(iris[, 1:4], k = 5)
```

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 list of class `"tidy_mds"` containing:

- config: tibble of MDS coordinates
- stress: `NA` (not applicable for classical MDS)
- gof: goodness-of-fit (proportion of variance retained)
- eigenvalues: numeric vector of eigenvalues
- method: `"Classical MDS"`
- model: the `cmdscale` result

Examples

```
d <- dist(USArrests)
mds <- tidy_mds_classical(d)
print(mds)
```

`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 list of class `"tidy_mds"` containing:

- config: tibble of MDS coordinates
- stress: Kruskal stress value
- method: `"Kruskal's isoMDS"`
- model: the `isoMDS` result

Examples

```
d <- dist(USArrests)
mds <- tidy_mds_kruskal(d)
```

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 list of class "tidy_mds" containing:

- config: tibble of MDS coordinates
- stress: Sammon stress value
- method: "Sammon Mapping"
- model: the sammon result

Examples

```
d <- dist(USArrests)
mds <- tidy_mds_sammon(d)
```

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 list of class "tidy_mds" containing:

- config: tibble of MDS coordinates
- stress: stress value from the SMACOF algorithm
- method: character string describing the MDS type
- model: the mds result

Examples

```
d <- dist(USArrests)
mds <- tidy_mds_smacof(d, type = "ratio")
```

---

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.

Examples

```
pca <- tidy_pca(USArrests)
tidy_pca_biplot(pca)
```

---

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.

Examples

```
pca <- tidy_pca(USArrests)
tidy_pca_screeplot(pca)
```

---

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 columns rule_id, lhs, rhs, and quality measures (e.g., support, confidence, lift).

Examples

```
if (requireNamespace("arules", quietly = TRUE)) {
  data("Groceries", package = "arules")
  rules_obj <- arules::apriori(Groceries,
    parameter = list(supp = 0.001, conf = 0.5))
  rules_tbl <- tidy_rules(rules_obj)
}
```

---

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

Examples

```
km <- kmeans(iris[, 1:4], centers = 3, nstart = 25)
d <- dist(iris[, 1:4])
sil <- tidy_silhouette(km$cluster, d)
```

`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 columns `k` and `avg_sil_width`. The `"optimal_k"` attribute contains the k with the highest average silhouette width.

Examples

```
sil_analysis <- tidy_silhouette_analysis(iris[, 1:4], max_k = 6)
```

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

The original data frame with an additional factor column named cluster_<method> containing cluster assignments. The fitted cluster model is stored as an attribute "cluster_model".

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 object with additional class "tidylearn_anomaly_aware". The model includes an anomaly_info element with anomaly_model, is_anomaly (logical vector), n_anomalies, and action.

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 object (class "tidylearn_model") fitted with the top significant interaction terms added to the formula. The interaction test results and selected interactions are stored as attributes "interaction_tests" and "selected_interactions".

Examples

```
model <- tl_auto_interactions(mtcars, mpg ~ wt + hp + cyl, top_n = 2)
```

---

tl_auto_ml *Auto ML: Automated Machine Learning Workflow*

Description

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

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). Controls which models are attempted and whether cross-validation is used for evaluation. The budget is checked **between** model fits, not during them – once a model starts training it runs to completion because R cannot safely interrupt C-level code (e.g. randomForest, xgboost, e1071). How the budget shapes the workflow: • **Under 30s**: Only fast models are attempted (tree, logistic/linear). Cross-validation is skipped; models are ranked on training-set metrics only. Expect 2 models in the leaderboard. Use this for quick sanity checks or interactive exploration. • **30–120s**: All baseline models are attempted including random forest. Cross-validation runs when enough time remains after each model fit; otherwise training metrics are used. Advanced models (SVM, XGBoost / ridge, lasso) are attempted if 40\ remains after baselines. Dimensionality reduction and clustering pipelines run if enabled and 10\ • **120s+ (recommended)**: The full pipeline runs – all baselines, advanced models, PCA-augmented variants, and cluster-augmented variants, each with cross-validation. Expect 9–11 models in the leaderboard. Because individual model fits (especially forest, SVM, XGBoost with CV) can take 5–30s each depending on data size, the actual wall-clock time may modestly exceed the budget by the duration of the last model that was started before the budget expired.
cv_folds	Number of cross-validation folds (default: 5). Reducing this (e.g. to 2 or 3) is an effective way to stay closer to the time budget since CV is typically the most expensive step.
metric	Evaluation metric (default: auto-selected based on task). For classification: "accuracy"; for regression: "rmse".

Value

A list with class `"tidylearn_automl"` containing:

best_model The best tidylearn model object

models Named list of all successfully trained models

leaderboard Tibble ranking models by the chosen metric

task Detected or specified task type

metric Metric used for ranking

runtime Total elapsed time as a difftime object

Examples

```
# Quick run with fast models only (< 30s budget skips forest/SVM/XGBoost)
result <- tl_auto_ml(iris, Species ~ .,
  time_budget = 10,
  use_reduction = FALSE,
  use_clustering = FALSE,
  cv_folds = 2)
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 with columns `metric` (character) and `value` (numeric) containing the requested classification metrics. When `thresholds` are supplied, additional rows are appended with threshold-specific metric names.

Examples

```
model <- tl_model(iris, Species ~ ., method = "forest")
preds <- predict(model)
tl_calc_classification_metrics(iris$Species, preds$.pred)
```

`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 named list with one element per assumption checked (`linearity`, `independence`, `homoscedasticity`, `normality`, `multicollinearity`, `outliers`), each containing `assumption` (character label), `check` (logical or `NULL`), `details` (character), and `recommendation` (character). An additional `overall` element summarises the number of assumptions checked, violated, and satisfied.

Examples

```
model <- tl_model(mtcars, mpg ~ wt + hp, method = "linear")
tl_check_assumptions(model)
```

`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 list with two elements:

`$fold_metrics` A data frame with columns `metric`, `value`, `fold`, and `model` containing per-fold results for every model.

`$summary` A data frame with columns `model`, `metric`, `mean_value`, `sd_value`, `min_value`, and `max_value` summarizing cross-validation performance.

Examples

```
m1 <- tl_model(mtcars, mpg ~ wt, method = "linear")
m2 <- tl_model(mtcars, mpg ~ wt + hp, method = "linear")
cv <- tl_compare_cv(mtcars, list(simple = m1, full = m2), folds = 3)
cv$summary
```

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 ggplot object showing a faceted bar chart comparing metric values across models, with the best model highlighted.

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

A list with two elements:

`$folds` A list of per-fold evaluation tibbles, each with `metric` and `value` columns.

`$summary` A tibble with columns `metric`, `mean`, and `sd` summarizing performance across folds.

Examples

```
cv <- tl_cv(mtcars, mpg ~ wt + hp, method = "linear", folds = 3)
cv$summary
```

`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 `shinyApp` object.

Examples

```
if (requireNamespace("shiny")) {
  model <- tl_model(mtcars, mpg ~ wt + hp, method = "linear")
  app <- tl_dashboard(model)
}
```

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 suitable for passing to tl_tune_grid or tl_tune_random. Each element is a numeric or character vector of candidate values for that hyperparameter.

Examples

```
grid <- tl_default_param_grid("tree", size = "small")
grid <- tl_default_param_grid("forest", size = "medium")
```

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:

method The detection method used (character).

method_name Human-readable method name (character).

threshold The threshold value used (numeric).

threshold_label Formatted threshold description (character).

outlier_flags A logical matrix (observations x variables).

any_outlier Logical vector indicating if each observation is an outlier in any variable.

outlier_counts List with `total`, `by_variable`, and `by_observation` counts.

outlier_indices Integer vector of outlier row indices.

plot A `ggplot` object, or `NULL` if `plot = FALSE`.

Examples

```
tl_detect_outliers(mtcars, variables = c("mpg", "wt"), method = "iqr")
```

---

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 grid.arrange object (a grob) containing the arranged diagnostic plots.

Examples

```
if (requireNamespace("gridExtra")) {
  model <- tl_model(mtcars, mpg ~ wt + hp, method = "linear")
  tl_diagnostic_dashboard(model)
}
```

---

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 with columns metric (character) and value (numeric). For regression models, includes rmse, mae, and rsq. For classification models, includes accuracy.

Examples

```
model <- tl_model(mtcars, mpg ~ wt + hp, method = "linear")
tl_evaluate(model)
```

`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

A list with class `"tidylearn_eda"` containing:

data The original data frame.

response The response variable name, or `NULL`.

pca The fitted PCA model.

optimal_k List with optimal cluster count results.

kmeans The fitted k-means model.

hclust The fitted hierarchical clustering model.

summary List with `n_obs`, `n_vars`, `n_components`, and `best_k`.

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` object from the pipeline, selected by the metric specified in `evaluation$best_metric`.

Examples

```
pipe <- tl_pipeline(iris, Species ~ .,
  models = list(tree = list(method = "tree")),
  evaluation = list(metrics = "accuracy", validation = "cv",
    cv_folds = 2, best_metric = "accuracy"))
pipe <- tl_run_pipeline(pipe, verbose = FALSE)
best <- tl_get_best_model(pipe)
```

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 one row per observation containing influence measures: cooks_distance, leverage, dffits, std_residual, stud_residual, boolean flags for each threshold (is_cook_influential, is_leverage_influential, is_dffits_influential, is_outlier), per-coefficient dfbetas_* columns, and an overall is_influential flag. Threshold values are stored as attributes.

Examples

```
model <- tl_model(mtcars, mpg ~ wt + hp, method = "linear")
tl_influence_measures(model)
```

---

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

For numeric var: a list with effects (data frame of predicted values across the variable range for each level of by_var) and slopes (data frame with estimated slopes and standard errors per level). For categorical var: a data frame of predicted values at each factor level for each level of by_var.

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 previously saved with tl_save_pipeline.

Examples

```
pipe <- tl_pipeline(iris, Species ~ .)
f <- tempfile(fileext = ".rds")
tl_save_pipeline(pipe, f)
pipe2 <- tl_load_pipeline(f)
```

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.

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 (S3) containing the fitted model (`$fit`), model specification (`$spec`), and training data (`$data`). The object also inherits from a method-specific class (e.g., `tidylearn_linear`) and a paradigm class (`tidylearn_supervised` or `tidylearn_unsupervised`).

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 (S3 list) with components `$formula`, `$data`, `$preprocessing`, `$models`, `$evaluation`, and `$results` (initially `NULL`; populated after `tl_run_pipeline`).

Examples

```
pipe <- tl_pipeline(iris, Species ~ .,
  models = list(tree = list(method = "tree")))
print(pipe)
```

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 showing boxplots of cross-validation metric distributions for each model.

Examples

```
m1 <- tl_model(mtcars, mpg ~ wt, method = "linear")
m2 <- tl_model(mtcars, mpg ~ wt + hp, method = "linear")
cv <- tl_compare_cv(mtcars, list(simple = m1, full = m2), folds = 3)
tl_plot_cv_comparison(cv)
```

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.

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

The return value of `keras::plot_model()`, an architecture diagram of the Keras model.

Examples

```
## Not run:
if (requireNamespace("keras", quietly = TRUE)) {
  model <- tl_model(iris, Species ~ ., method = "deep", epochs = 5)
  tl_plot_deep_architecture(model)
}

## End(Not run)
```

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.

Examples

```
## Not run:
if (requireNamespace("keras", quietly = TRUE)) {
  model <- tl_model(iris, Species ~ ., method = "deep", epochs = 5)
  tl_plot_deep_history(model)
}

## End(Not run)
```

`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.

Examples

```
iris_bin <- iris[iris$Species != "setosa", ]
iris_bin$Species <- factor(iris_bin$Species)
model <- tl_model(iris_bin, Species ~ ., method = "logistic")
tl_plot_gain(model)
```

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.

Examples

```
m1 <- tl_model(iris, Species ~ ., method = "forest")
m2 <- tl_model(iris, Species ~ ., method = "boost")
tl_plot_importance_comparison(m1, m2, names = c("Forest", "Boost"))
```

tl_plot_importance_regularized

Plot variable importance for a regularized model

Description

Plot variable importance for a regularized 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.

Examples

```
model <- tl_model(mtcars, mpg ~ ., method = "lasso")
tl_plot_importance_regularized(model)
```

`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.

Examples

```
model <- tl_model(mtcars, mpg ~ wt + hp, method = "linear")
tl_plot_influence(model, plot_type = "cook")
```

`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.

Examples

```
model <- tl_model(mtcars, mpg ~ wt, method = "linear")
tl_plot_intervals(model)
```

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.

Examples

```
iris_bin <- iris[iris$Species != "setosa", ]
iris_bin$Species <- factor(iris_bin$Species)
model <- tl_model(iris_bin, Species ~ ., method = "logistic")
tl_plot_lift(model)
```

---

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.

Examples

```
m1 <- tl_model(mtcars, mpg ~ wt + hp, method = "linear")
m2 <- tl_model(mtcars, mpg ~ wt + hp, method = "lasso")
tl_plot_model_comparison(m1, m2, names = c("Linear", "Lasso"))
```

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

The return value of plotnet, called for its side effect of drawing the network diagram, or NULL if the **NeuralNetTools** package is not installed.

Examples

```
if (requireNamespace("NeuralNetTools", quietly = TRUE)) {
  model <- tl_model(iris, Species ~ ., method = "nn", size = 3)
  tl_plot_nn_architecture(model)
}
```

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.

`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.

Examples

```
model <- tl_model(mtcars, mpg ~ ., method = "forest")
tl_plot_partial_dependence(model, var = "wt")
```

`tl_plot_regularization_cv`

Plot cross-validation results for a regularized 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.

Examples

```
model <- tl_model(mtcars, mpg ~ ., method = "ridge")
tl_plot_regularization_cv(model)
```

---

tl_plot_regularization_path

Plot regularization path for a regularized model

Description

Plot regularization path for a regularized 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.

Examples

```
model <- tl_model(mtcars, mpg ~ ., method = "lasso")
tl_plot_regularization_path(model)
```

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.

Examples

```
if (requireNamespace("e1071", quietly = TRUE)) {
  model <- tl_model(iris, Species ~ ., method = "svm")
  tl_plot_svm_boundary(model,
    x_var = "Sepal.Length", y_var = "Sepal.Width")
}
```

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.

Examples

```
if (requireNamespace("e1071", quietly = TRUE)) {
  model <- tl_model(iris, Species ~ ., method = "svm",
    kernel = "linear", tune = TRUE, tune_folds = 2)
  tl_plot_svm_tuning(model)
}
```

---

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

The return value of rpart.plot, called for its side effect of drawing the tree.

Examples

```
model <- tl_model(iris, Species ~ ., method = "tree")
tl_plot_tree(model)
```

`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.

Examples

```
model <- tl_tune_grid(iris, Species ~ ., method = "tree",
  param_grid = list(cp = c(0.01, 0.1), minsplit = c(10, 20)),
  folds = 2, verbose = FALSE)
tl_plot_tuning_results(model)
```

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.

Examples

```
if (requireNamespace("xgboost", quietly = TRUE)) {
  model <- tl_model(mtcars, mpg ~ ., method = "xgboost")
  tl_plot_xgboost_importance(model)
}
```

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.

---

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.

Examples

```
if (requireNamespace("xgboost", quietly = TRUE)) {
  model <- tl_model(mtcars, mpg ~ ., method = "xgboost")
  tl_plot_xgboost_shap_summary(model, n_samples = 20)
}
```

`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

The return value of `xgb.plot.tree`, a tree diagram rendered via the **DiagrammeR** package.

`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

A tibble with a .pred column containing predictions from the selected (or best) pipeline model, after applying the same preprocessing steps used during training.

---

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 with components:

`data` The processed data frame.

`original_data` The original unprocessed data frame.

`preprocessing_steps` A list of metadata for each preprocessing step applied (imputation values, encoding maps, scaling parameters, etc.).

`formula` The formula passed in (or `NULL`).

Examples

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

---

tl_read *Read data from diverse sources*

Description

Auto-detects the data format from the file extension or source pattern and dispatches to the appropriate reader. All readers return a `tidylearn_data` object, which is a tibble subclass carrying metadata about the data source.

Usage

```
tl_read(source, ..., format = NULL, .quiet = FALSE)
```

Arguments

`source`	A file path, URL, connection string, directory path, or a character vector of multiple file paths.
`...`	Additional arguments passed to the format-specific reader.
`format`	Optional explicit format override. One of `"csv"`, `"tsv"`, `"excel"`, `"parquet"`, `"json"`, `"rds"`, `"rdata"`, `"sqlite"`, `"postgres"`, `"mysql"`, `"bigquery"`, `"s3"`, `"github"`, `"kaggle"`. When `NULL` (default), the format is auto-detected from the file extension or source pattern. Note: `.txt` files default to CSV; use `format = "tsv"` to override.
`.quiet`	Logical. If `TRUE`, suppresses informational messages. Default is `FALSE`.

Details

When `source` is a character vector of multiple paths, each file is read and row-bound into a single result with a `source_file` column. When `source` is a directory path, it is equivalent to calling `tl_read_dir()`. When `source` is a `.zip` file, it is equivalent to calling `tl_read_zip()`.

Value

A `tidylearn_data` object (a tibble subclass) with attributes `tl_source`, `tl_format`, and `tl_timestamp`.

Examples

```
# Read a single CSV file
# data <- tl_read("path/to/data.csv")

# Read multiple files and row-bind
# data <- tl_read(c("jan.csv", "feb.csv", "mar.csv"))

# Read all CSVs from a directory
# data <- tl_read("data/")

# Read from a zip archive
# data <- tl_read("data.zip")

# Explicit format override
# data <- tl_read("path/to/data.txt", format = "tsv")
```

`tl_read_bigquery` *Read from Google BigQuery*

Description

Executes a SQL query against Google BigQuery and returns the result as a `tidylearn_data` object. Requires the **bigrquery** package and valid Google Cloud authentication.

Usage

```
tl_read_bigquery(project, query, dataset = NULL, ...)
```

Arguments

`project`	Google Cloud project ID.
`query`	A SQL query string (Standard SQL).
`dataset`	Optional default dataset for unqualified table names.
`...`	Additional arguments passed to `bigrquery::bq_project_query()`.

Value

A `tidylearn_data` object containing the query results.

Examples

```
# data <- tl_read_bigquery(
#   project = "my-project",
#   query = "SELECT * FROM `my_dataset.my_table` LIMIT 1000"
# )
```

`tl_read_csv` *Read a CSV file*

Description

Reads a CSV file into a `tidylearn_data` object. Uses **readr** when available for faster parsing, with a base R fallback.

Usage

```
tl_read_csv(path, ...)
```

Arguments

`path`	Path to a CSV file.
`...`	Additional arguments passed to `readr::read_csv()` or `utils::read.csv()`.

Value

A `tidylearn_data` object (a tibble subclass) with attributes `tl_source`, `tl_format`, and `tl_timestamp`.

Examples

```
# data <- tl_read_csv("path/to/data.csv")
```

tl_read_db *Read from a DBI database connection*

Description

Executes a SQL query against an existing **DBI** connection and returns the result as a `tidylearn_data` object. The connection is not closed by this function — the caller is responsible for managing the connection lifecycle.

Usage

```
tl_read_db(conn, query, ...)
```

Arguments

`conn`	A **DBI** connection object (e.g., from `DBI::dbConnect()`).
`query`	A SQL query string.
`...`	Additional arguments passed to `DBI::dbGetQuery()`.

Value

A `tidylearn_data` object containing the query results.

Examples

```
# conn <- DBI::dbConnect(RSQLite::SQLite(), "my_database.sqlite")
# data <- tl_read_db(conn, "SELECT * FROM my_table")
# DBI::dbDisconnect(conn)
```

tl_read_dir *Read all matching files from a directory*

Description

Scans a directory for files matching a pattern or format, reads each one, and row-binds them into a single `tidylearn_data` object with a `source_file` column identifying the origin of each row.

Usage

```
tl_read_dir(
  path,
  pattern = NULL,
  format = NULL,
  recursive = FALSE,
  .quiet = FALSE,
  ...
)
```

Arguments

path	Path to a directory.
pattern	Optional regex pattern to filter file names (e.g., "sales_.*\\.csv$"). If NULL, files are filtered by format instead.
format	File format to read. If NULL and pattern is NULL, all recognized data files are read. If specified, only files with matching extensions are read.
recursive	Logical. Should subdirectories be scanned? Default is FALSE.
.quiet	Suppress messages. Default is FALSE.
...	Additional arguments passed to the format-specific reader.

Value

A tidylearn_data object with an additional source_file column identifying the origin of each row.

Examples

```
# Read all CSVs from a directory
# data <- tl_read_dir("data/", format = "csv")

# Read with pattern matching
# data <- tl_read_dir("data/", pattern = "^sales_.*\.csv$")

# Read all recognized data files recursively
# data <- tl_read_dir("data/", recursive = TRUE)
```

---

tl_read_excel *Read an Excel file*

Description

Reads an Excel file (.xls, .xlsx, or .xlsm) into a tidylearn_data object. Requires the **readxl** package.

Usage

```
tl_read_excel(path, sheet = 1, ...)
```

Arguments

path	Path to an Excel file.
sheet	Sheet to read. Either a string (the name of a sheet) or an integer (the position of the sheet). Defaults to the first sheet.
...	Additional arguments passed to `readxl::read_excel()`.

Value

A `tidylearn_data` object (a tibble subclass) with attributes `tl_source`, `tl_format`, and `tl_timestamp`.

Examples

```
# data <- tl_read_excel("path/to/data.xlsx")
# data <- tl_read_excel("path/to/data.xlsx", sheet = "Sheet2")
```

---

tl_read_github *Read from GitHub*

Description

Downloads a raw file from a GitHub repository and reads it into a `tidylearn_data` object. Accepts either a full GitHub URL or a `owner/repo` shorthand with a file path.

Usage

```
tl_read_github(source, path = NULL, ref = "main", ...)
```

Arguments

source	A GitHub URL or `"owner/repo"` string.
path	Path to the file within the repository (required when `source` is `"owner/repo"` format).
ref	Branch, tag, or commit SHA. Default is `"main"`.
...	Additional arguments passed to the format-specific reader.

Value

A `tidylearn_data` object containing the downloaded data.

Examples

```
# data <- tl_read_github("user/repo", path = "data/file.csv")
# data <- tl_read_github(
#   "https://github.com/user/repo/blob/main/data/file.csv"
# )
```

`tl_read_json` *Read a JSON file*

Description

Reads a JSON file into a `tidylearn_data` object. Expects the JSON to represent tabular data (array of objects or similar). Requires the **jsonlite** package.

Usage

```
tl_read_json(path, flatten = TRUE, ...)
```

Arguments

`path`	Path to a JSON file.
`flatten`	Logical. Automatically flatten nested data frames? Default is `TRUE`.
`...`	Additional arguments passed to `jsonlite::fromJSON()`.

Value

A `tidylearn_data` object (a tibble subclass) with attributes `tl_source`, `tl_format`, and `tl_timestamp`.

Examples

```
# data <- tl_read_json("path/to/data.json")
```

tl_read_kaggle *Read from Kaggle*

Description

Downloads a dataset file from Kaggle using the Kaggle CLI and reads it into a `tidylearn_data` object. Requires the Kaggle CLI to be installed and configured (`pip install kaggle`).

Usage

```
tl_read_kaggle(source, file = NULL, dest = tempdir(), type = "dataset", ...)
```

Arguments

`source`	A Kaggle dataset slug (e.g., `"user/dataset-name"`) or a Kaggle URL.
`file`	The specific file to read from the dataset. If `NULL` and the dataset contains exactly one file, it is read automatically.
`dest`	Directory to download files to. Default is a temporary directory.
`type`	Either `"dataset"` (default) or `"competition"`.
`...`	Additional arguments passed to the format-specific reader.

Value

A `tidylearn_data` object containing the downloaded data.

Examples

```
# data <- tl_read_kaggle("zillow/zecon", file = "Zip_time_series.csv")
# data <- tl_read_kaggle("titanic", file = "train.csv", type = "competition")
```

tl_read_mysql *Read from a MySQL/MariaDB database*

Description

Connects to a MySQL or MariaDB database, executes a SQL query, and returns the result as a `tidylearn_data` object. Accepts either a connection string or individual connection parameters. Requires **DBI** and **RMariaDB**.

Usage

```
tl_read_mysql(
  dsn,
  query,
  dbname = NULL,
  user = NULL,
  password = NULL,
  port = 3306,
  ...
)
```

Arguments

dsn	A MySQL connection string (e.g., "mysql://user:pass@host:port/dbname"), or the database host if using named parameters.
query	A SQL query string.
dbname	Database name (if not in dsn).
user	Username (if not in dsn).
password	Password (if not in dsn).
port	Port number. Default is 3306.
...	Additional arguments passed to DBI::dbConnect().

Value

A tidylearn_data object containing the query results.

Examples

```
# data <- tl_read_mysql(
#   dsn = "localhost",
#   query = "SELECT * FROM my_table",
#   dbname = "mydb",
#   user = "myuser",
#   password = "mypass"
# )
```

---

tl_read_parquet *Read a Parquet file*

Description

Reads a Parquet file into a tidylearn_data object. Requires the **nanoparquet** package.

Usage

```
tl_read_parquet(path, ...)
```

Arguments

path	Path to a Parquet file.
...	Additional arguments passed to nanoparquet::read_parquet().

Value

A tidylearn_data object (a tibble subclass) with attributes tl_source, tl_format, and tl_timestamp.

Examples

```
# data <- tl_read_parquet("path/to/data.parquet")
```

---

tl_read_postgres *Read from a PostgreSQL database*

Description

Connects to a PostgreSQL database, executes a SQL query, and returns the result as a tidylearn_data object. Accepts either a connection string or individual connection parameters. Requires **DBI** and **RPostgres**.

Usage

```
tl_read_postgres(
  dsn,
  query,
  dbname = NULL,
  user = NULL,
  password = NULL,
  port = 5432,
  ...
)
```

Arguments

dsn	A PostgreSQL connection string (e.g., "postgres://user:pass@host:port/dbname"), or the database host if using named parameters.
query	A SQL query string.
dbname	Database name (if not in dsn).
user	Username (if not in dsn).

password	Password (if not in dsn).
port	Port number. Default is 5432.
...	Additional arguments passed to DBI::dbConnect().

Value

A tidylearn_data object containing the query results.

Examples

```
# data <- tl_read_postgres(
#   dsn = "localhost",
#   query = "SELECT * FROM my_table",
#   dbname = "mydb",
#   user = "myuser",
#   password = "mypass"
# )
```

---

tl_read_rdata *Read an RData file*

Description

Reads an RData (.rdata or .rda) file into a tidylearn_data object. Since RData files can contain multiple objects, use the name argument to specify which object to extract. If name is NULL and the file contains exactly one data frame, it is returned automatically.

Usage

```
tl_read_rdata(path, name = NULL, ...)
```

Arguments

path	Path to an RData file.
name	Optional name of the object to extract from the RData file. If NULL (default), the function returns the first data frame found, or errors if there are multiple data frames.
...	Currently unused.

Value

A tidylearn_data object (a tibble subclass) with attributes tl_source, tl_format, and tl_timestamp.

Examples

```
# data <- tl_read_rdata("path/to/data.rdata")
# data <- tl_read_rdata("path/to/data.rdata", name = "my_data")
```

---

tl_read_rds *Read an RDS file*

Description

Reads an RDS file into a `tidylearn_data` object. Uses base R `readRDS()` — no additional packages required.

Usage

```
tl_read_rds(path)
```

Arguments

`path`	Path to an RDS file.

Value

A `tidylearn_data` object (a tibble subclass) with attributes `tl_source`, `tl_format`, and `tl_timestamp`.

Examples

```
# data <- tl_read_rds("path/to/data.rds")
```

---

tl_read_s3 *Read from Amazon S3*

Description

Downloads a file from an S3 bucket and reads it into a `tidylearn_data` object. The file format is auto-detected from the key's extension, or can be specified explicitly. Requires the **paws.storage** package and valid AWS credentials.

Usage

```
tl_read_s3(source, format = NULL, region = NULL, ...)
```

Arguments

source	An S3 URI (e.g., "s3://bucket/path/to/file.csv").
format	Optional format override for the downloaded file. If NULL, auto-detected from the S3 key extension.
region	AWS region. If NULL, uses the default from your AWS configuration.
...	Additional arguments passed to the format-specific reader.

Value

A tidylearn_data object containing the downloaded data.

Examples

```
# data <- tl_read_s3("s3://my-bucket/data/sales.csv")
# data <- tl_read_s3("s3://my-bucket/data/results.parquet")
```

---

tl_read_sqlite *Read from a SQLite database*

Description

Opens a SQLite database file, executes a SQL query, and returns the result as a tidylearn_data object. The connection is automatically closed when done. Requires **DBI** and **RSQLite**.

Usage

```
tl_read_sqlite(path, query, ...)
```

Arguments

path	Path to a SQLite database file (.sqlite or .db).
query	A SQL query string.
...	Additional arguments passed to DBI::dbGetQuery().

Value

A tidylearn_data object containing the query results.

Examples

```
# data <- tl_read_sqlite("my_database.sqlite", "SELECT * FROM my_table")
```

tl_read_tsv *Read a TSV file*

Description

Reads a tab-separated file into a `tidylearn_data` object. Uses **readr** when available for faster parsing, with a base R fallback.

Usage

```
tl_read_tsv(path, ...)
```

Arguments

`path`	Path to a TSV file.
`...`	Additional arguments passed to `readr::read_tsv()` or `utils::read.delim()`.

Value

A `tidylearn_data` object (a tibble subclass) with attributes `tl_source`, `tl_format`, and `tl_timestamp`.

Examples

```
# data <- tl_read_tsv("path/to/data.tsv")
```

tl_read_zip *Read data from a zip archive*

Description

Extracts a zip archive to a temporary directory and reads the contents. If the archive contains a single data file, it is read directly. If multiple data files are found, they are row-bound with a `source_file` column. Use the `file` argument to select a specific file from the archive.

Usage

```
tl_read_zip(path, file = NULL, format = NULL, .quiet = FALSE, ...)
```

Arguments

`path`	Path to a zip file.
`file`	Optional name of a specific file within the archive to read. Supports partial matching.
`format`	Optional format override for the file(s) inside the archive.
`.quiet`	Suppress messages. Default is `FALSE`.
`...`	Additional arguments passed to the format-specific reader.

Value

A `tidylearn_data` object (a tibble subclass) with attributes `tl_source`, `tl_format`, and `tl_timestamp`. The archive is extracted to a temporary directory that is cleaned up automatically. If multiple data files are found, a `source_file` column identifies the origin of each row.

Examples

```
# Read from a zip archive
# data <- tl_read_zip("data.zip")

# Read a specific file from the archive
# data <- tl_read_zip("data.zip", file = "train.csv")
```

---

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 with components:

data The transformed data frame with reduced-dimension columns and the response variable (if provided).

reduction_model The fitted tidylearn dimensionality reduction model.

original_data The original input data frame.

response The response variable name, or NULL.

Examples

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

---

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

The input tidylearn_pipeline object with its $results component populated. Results include $processed_data, $model_results (a named list of per-model fits and metrics), $best_model_name, $best_model (the winning tidylearn_model), and $metric_values.

Examples

```
pipe <- tl_pipeline(iris, Species ~ .,
  models = list(tree = list(method = "tree")),
  evaluation = list(metrics = "accuracy", validation = "cv",
    cv_folds = 2, best_metric = "accuracy"))
pipe <- tl_run_pipeline(pipe, verbose = FALSE)
```

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

Called for its side effect of saving to disk; returns NULL invisibly.

Examples

```
pipe <- tl_pipeline(iris, Species ~ .)
tl_save_pipeline(pipe, tempfile(fileext = ".rds"))
```

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 object with additional class "tidylearn_semisupervised", trained on pseudo-labeled data. The model includes a semisupervised_info element with labeled_indices, cluster_model, and label_mapping.

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 = "tree"
)
```

---

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 two elements:

`$train` A data frame containing the training subset.

`$test` A data frame containing the test subset.

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 `tidylearn_model` object of class `tidylearn_linear` wrapping the selected `lm` model. Access the underlying model via `$fit` and the selected formula via `$spec$formula`.

Examples

```
model <- tl_step_selection(mtcars, mpg ~ ., direction = "backward")
summary(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 with class `"tidylearn_stratified"` containing:

cluster_model The fitted clustering model.

supervised_models Named list of tidylearn models, one per cluster.

formula The model formula.

data The original training data.

Examples

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

tl_table *Create formatted tables for tidylearn models*

Description

Dispatches to the appropriate table function based on model type and requested table type. Requires the gt package.

Usage

```
tl_table(model, type = "auto", ...)
```

Arguments

model	A tidylearn model object
type	Table type (default: "auto"). For supervised models: "metrics", "coefficients", "confusion", "importance". For unsupervised models: "variance", "loadings", "clusters". MDS models are not supported.
...	Additional arguments passed to the underlying table function

Value

A gt table object.

Examples

```
model <- tl_model(mtcars, mpg ~ wt + hp, method = "linear")
tl_table(model)
tl_table(model, type = "coefficients")
```

tl_table_clusters *Formatted cluster summary table*

Description

Produces a styled gt table showing cluster sizes and mean feature values. Supports kmeans, pam, clara, dbscan, and hclust models.

Usage

```
tl_table_clusters(model, k = 3, digits = 2, ...)
```

Arguments

model	A tidylearn clustering model object
k	For hclust models, the number of clusters to cut (default: 3)
digits	Number of decimal places (default: 2)
...	Additional arguments (currently unused)

Value

A gt table object.

Examples

```
model <- tl_model(iris[, 1:4], method = "kmeans", k = 3)
tl_table_clusters(model)
```

---

tl_table_coefficients *Formatted model coefficients table*

Description

Produces a styled gt table of model coefficients. Supports linear, polynomial, logistic, ridge, lasso, and elastic net models.

Usage

```
tl_table_coefficients(model, lambda = "1se", digits = 4, ...)
```

Arguments

model	A tidylearn model object
lambda	For regularised models: "1se" (default) or "min"
digits	Number of decimal places (default: 4)
...	Additional arguments (currently unused)

Value

A gt table object.

Examples

```
model <- tl_model(mtcars, mpg ~ wt + hp, method = "linear")
tl_table_coefficients(model)
```

tl_table_comparison *Compare multiple models in a formatted table*

Description

Evaluates multiple tidylearn models and presents the results side-by-side in a styled gt table.

Usage

```
tl_table_comparison(..., new_data = NULL, names = NULL, digits = 4)
```

Arguments

...	tidylearn model objects to compare
new_data	Optional test data for evaluation. If NULL, uses the training data of the first model.
names	Optional character vector of model names
digits	Number of decimal places (default: 4)

Value

A gt table object.

Examples

```
m1 <- tl_model(mtcars, mpg ~ ., method = "linear")
m2 <- tl_model(mtcars, mpg ~ ., method = "lasso")
tl_table_comparison(m1, m2, names = c("Linear", "Lasso"))
```

tl_table_confusion *Formatted confusion matrix table*

Description

Produces a styled gt confusion matrix with correct predictions highlighted. Only available for classification models.

Usage

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

Arguments

model	A tidylearn classification model
new_data	Optional test data. If NULL, uses training data.
...	Additional arguments (currently unused)

Value

A gt table object.

Examples

```
model <- tl_model(iris, Species ~ ., method = "forest")
tl_table_confusion(model)
```

---

tl_table_importance *Formatted feature importance table*

Description

Produces a styled gt table of feature importance with a colour gradient. Supports tree-based, regularised, and xgboost models.

Usage

```
tl_table_importance(model, top_n = 20, digits = 2, ...)
```

Arguments

model	A tidylearn model object
top_n	Maximum number of features to display (default: 20)
digits	Number of decimal places (default: 2)
...	Additional arguments (currently unused)

Value

A gt table object.

Examples

```
model <- tl_model(iris, Species ~ ., method = "forest")
tl_table_importance(model)
```

tl_table_loadings *Formatted PCA loadings table*

Description

Produces a styled gt table of variable loadings on each principal component, with a diverging colour scale to highlight strong loadings.

Usage

```
tl_table_loadings(model, n_components = NULL, digits = 3, ...)
```

Arguments

model	A tidylearn PCA model object
n_components	Number of components to show (default: all)
digits	Number of decimal places (default: 3)
...	Additional arguments (currently unused)

Value

A gt table object.

Examples

```
model <- tl_model(iris[, 1:4], method = "pca")
tl_table_loadings(model)
```

tl_table_metrics *Formatted evaluation metrics table*

Description

Produces a styled gt table of model evaluation metrics from tl_evaluate.

Usage

```
tl_table_metrics(model, new_data = NULL, digits = 4, ...)
```

Arguments

model	A tidylearn supervised model object
new_data	Optional test data. If NULL, uses training data.
digits	Number of decimal places (default: 4)
...	Additional arguments passed to tl_evaluate

Value

A gt table object.

Examples

```
model <- tl_model(mtcars, mpg ~ wt + hp, method = "linear")
tl_table_metrics(model)
```

`tl_table_variance` *Formatted PCA variance explained table*

Description

Produces a styled gt table of variance explained by each principal component, with a colour gradient on cumulative variance.

Usage

```
tl_table_variance(model, n_components = NULL, digits = 4, ...)
```

Arguments

`model`	A tidylearn PCA model object
`n_components`	Maximum number of components to show (default: all)
`digits`	Number of decimal places (default: 4)
`...`	Additional arguments (currently unused)

Value

A gt table object.

Examples

```
model <- tl_model(iris[, 1:4], method = "pca")
tl_table_variance(model)
```

`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 one row per tested interaction pair, containing columns `var1`, `var2`, `p_value`, `significant` (logical), `delta_r2` (change in R-squared), and `f_statistic`, sorted by `p_value` ascending.

Examples

```
results <- tl_test_interactions(mtcars, mpg ~ wt + hp + cyl,
  var1 = "wt", var2 = "hp")
```

`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 columns `metric`, `model`, `baseline`, `mean_diff`, `p_value`, and `p_adj` (Holm-adjusted p-value) containing pairwise statistical comparisons against the baseline model.

Examples

```
m1 <- tl_model(mtcars, mpg ~ wt, method = "linear")
m2 <- tl_model(mtcars, mpg ~ wt + hp, method = "linear")
cv <- tl_compare_cv(mtcars, list(simple = m1, full = m2), folds = 3)
tl_test_model_difference(cv, baseline_model = "simple", metric = "rmse")
```

`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 list with class `"tidylearn_transfer"` containing:

pretrain_model The fitted dimensionality reduction model.

supervised_model The fitted supervised tidylearn model.

formula The model formula.

method The supervised learning method used.

Examples

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

`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 elements `model` (the best fitted deep learning model), `best_hidden_layers` (optimal layer configuration), `best_learning_rate`, `best_batch_size`, and `tuning_results` (a data frame of all hyperparameter combinations and their validation losses).

Examples

```
## Not run:
if (requireNamespace("keras", quietly = TRUE)) {
  result <- tl_tune_deep(iris, Species ~ .,
    is_classification = TRUE,
    hidden_layers_options = list(c(10), c(10, 5)),
    learning_rates = c(0.01, 0.001), batch_sizes = c(32),
    epochs = 5)
}

## End(Not run)
```

---

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 tidylearn model object fitted with the best hyperparameters. Tuning results are stored as an attribute "tuning_results", a list containing param_grid, results (data frame of all evaluated combinations), best_params, best_metric, metric, and maximize.

Examples

```
model <- tl_tune_grid(iris, Species ~ ., method = "tree",
  param_grid = list(cp = c(0.01, 0.1), minsplit = c(10, 20)),
  folds = 2, verbose = FALSE)
```

---

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 elements model (the best fitted nnet model), best_size (optimal hidden-layer size), best_decay (optimal weight decay), and tuning_results (a data frame of all parameter combinations and their cross-validated errors).

`tl_tune_random` *Tune hyperparameters using random search*

Description

Tune hyperparameters 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 tidylearn model object fitted with the best hyperparameters. Tuning results are stored as an attribute `"tuning_results"`, a list containing `param_space`, `results` (data frame of all evaluated iterations), `best_params`, `best_metric`, `metric`, and `maximize`.

Examples

```
model <- tl_tune_random(mtcars, mpg ~ ., method = "tree",
  param_space = list(cp = c(0.01, 0.1), minsplit = c(10, 20)),
  n_iter = 3, folds = 2, verbose = FALSE)
```

`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 `tidylearn_model` object (the refit on full data using the best hyperparameters) with an attribute `"tuning_results"` containing a list with elements `param_grid`, `results` (per-combination CV output), `best_params`, `best_iteration`, `best_score`, and `minimize`.

tl_version *Get tidylearn version information*

Description

Get tidylearn version information

Usage

```
tl_version()
```

Value

A package_version object containing the version number

Examples

```
tl_version()
```

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 one column of SHAP values per feature, a BIAS column, a row_id column, and the original data columns appended for reference.

Examples

```
if (requireNamespace("xgboost", quietly = TRUE)) {
  model <- tl_model(mtcars, mpg ~ ., method = "xgboost")
  shap <- tl_xgboost_shap(model, n_samples = 20)
}
```

`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

A `ggplot` object when `method = "scatter"`. For other methods, the plot is produced as a side effect via **arulesViz**.

Examples

```
if (requireNamespace("arules", quietly = TRUE)) {
  data("Groceries", package = "arules")
  res <- tidy_apriori(Groceries, support = 0.001, confidence = 0.5)
  visualize_rules(res, method = "scatter")
}
```

Index