

Package: limestab (via r-universe)

October 15, 2026

Type Package

Title Stable Local Interpretable Model-Agnostic Explanations (LIME, OptiLIME, BayLIME, ALIME)

Version 0.1.0

Description Provides four local explanation methods for tabular data: LIME, OptiLIME, BayLIME, and ALIME, together with the stability metrics used to compare them across repeated explanations of the same instance: the Coefficient Stability Index (CSI), the Variable Stability Index (VSI), Median Rank Matching Stability (MRMS), Point-Based Stability Index (PCSI), and Interval Coefficient Stability Index (ICSI). The methodology accompanies Toparlak and Purutcuoglu, ``A Median-Based Framework for Assessing the Stability in XAI Tools" (manuscript under revision); the citation will be updated upon publication.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 3.6)

Imports stats, utils

Suggests xgboost, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Simay Toparlak [aut] (affiliation: Department of Biomedical Engineering, Middle East Technical University), Vilda Purutcuoglu [aut, cre] (affiliation: Department of Statistics, Middle East Technical University)

Maintainer Vilda Purutcuoglu <vpurutcu@metu.edu.tr>

Repository <https://cran.r-universe.dev>

Date/Publication 2026-10-15 16:30:12 UTC

RemoteUrl <https://github.com/cran/limestab>

RemoteRef HEAD

RemoteSha bafba625b4d72e0b4bd54c7ac34f57c5c7e49886

Contents

limestab-package	2
alime_explain_instance	3
alime_explainer	3
bayesian_optimisation	4
bayesian_ridge_fit	5
baylime_explain_instance	6
baylime_explainer	7
calculate_distances	8
compute_cov_metrics	8
compute_csi	9
compute_icsi	9
compute_mad_stdevs	10
compute_mrms	10
compute_pcsi	11
compute_vsi	11
compute_wls_stdevs	12
default_kernel_width	12
fit_gp	13
fit_weighted_ridge	13
generate_perturbations	14
lime_check_stability	14
lime_confidence_intervals	15
lime_explain_instance	16
lime_explainer	17
lime_kernel	17
optilime	18
optilime_score	19
predict.optilime_gp	19
select_features	20
weighted_r2	21
Index	22

limestab-package	<i>limestab: Stable LIME, OptiLIME, BayLIME and ALIME</i>
------------------	---

Description

Tools for local explanations of tabular black-box models using LIME, OptiLIME, BayLIME and ALIME, together with stability diagnostics for repeated explanations.

Details

The package provides coefficient, variable-selection, ranking, point-based and interval-based stability measures for repeated local explanations.

`alime_explain_instance`*Explain a single instance with ALIME*

Description

Explain a single instance with ALIME.

Usage

```
alime_explain_instance(explainer,  
                        data_row,  
                        predict_fn,  
                        label = 2,  
                        num_features = 5,  
                        num_samples = 10000,  
                        seed = NULL)
```

Arguments

<code>explainer</code>	An <code>alime_explainer</code> object.
<code>data_row</code>	The instance to explain.
<code>predict_fn</code>	Prediction function.
<code>label</code>	Column of <code>predict_fn</code> 's output to explain.
<code>num_features</code>	Number of features kept in the local model.
<code>num_samples</code>	Number of points perturbed before keeping the nearest <code>n_neighbors</code> .
<code>seed</code>	Optional seed for reproducibility.

Value

A list with the same structure as `lime_explain_instance()`.

`alime_explainer`*Create an ALIME explainer*

Description

Create an ALIME explainer.

Usage

```
alime_explainer(training_data,
  feature_names = NULL,
  n_neighbors = 300,
  feature_selection = "forward_selection",
  penalty = 1,
  standardize = TRUE,
  embed_fn = NULL,
  random_state = NULL)
```

Arguments

<code>training_data</code>	Numeric training-data matrix.
<code>feature_names</code>	Optional character vector of feature names.
<code>n_neighbors</code>	Number of nearest perturbed points kept for the local model.
<code>feature_selection</code>	Feature-selection method.
<code>penalty</code>	Ridge penalty of the local model.
<code>standardize</code>	If TRUE, measure ALIME neighbourhood distances in the standardised feature space.
<code>embed_fn</code>	Optional embedding function applied before measuring distances.
<code>random_state</code>	Optional default seed.

Value

An object of class `alime_explainer`.

`bayesian_optimisation` *Bayesian optimisation of a scalar objective*

Description

Bayesian optimisation of a scalar objective.

Usage

```
bayesian_optimisation(n_iters,
  sample_loss,
  bounds,
  n_pre_samples = 5,
  random_state = 42,
  verbose = FALSE)
```

Arguments

n_iters	Number of Bayesian-optimisation iterations after the initial design.
sample_loss	Function of one scalar returning the value to maximise.
bounds	c(lower, upper) search bounds.
n_pre_samples	Number of random initial evaluations.
random_state	Seed for the whole procedure.
verbose	If TRUE, print progress.

Value

A list with x_sample and y_sample, every point evaluated.

bayesian_ridge_fit	<i>Bayesian ridge regression (Tipping 2001 / MacKay 1992)</i>
--------------------	---

Description

Bayesian ridge regression (Tipping 2001 / MacKay 1992).

Usage

```
bayesian_ridge_fit(x,
  y,
  w,
  alpha_init = NULL,
  lambda_init = NULL,
  fit_alpha = TRUE,
  fit_lambda = TRUE,
  max_iter = 1000,
  tol = 1e-4,
  alpha_1 = 1e-6,
  alpha_2 = 1e-6,
  lambda_1 = 1e-6,
  lambda_2 = 1e-6,
  weighted_centering = FALSE)
```

Arguments

x	Numeric predictor matrix.
y	Numeric response vector.
w	Sample weights.
alpha_init	Initial (or fixed) noise precision; NULL uses 1 / var(y).
lambda_init	Initial (or fixed) weight precision; NULL uses 1.
fit_alpha	Whether the noise precision is re-estimated from data.

fit_lambda	Whether the weight precision is re-estimated from data.
max_iter	Maximum number of update iterations.
tol	Convergence tolerance on the coefficients.
alpha_1	Gamma-prior shape hyperparameter for alpha.
alpha_2	Gamma-prior rate hyperparameter for alpha.
lambda_1	Gamma-prior shape hyperparameter for lambda.
lambda_2	Gamma-prior rate hyperparameter for lambda.
weighted_centering	If TRUE, centre x/y with weighted means; if FALSE, use unweighted means.

Value

A list with intercept, coef, sigma (posterior covariance), alpha and lambda.

baylime_explain_instance
Explain a single instance with BayLIME

Description

Explain a single instance with BayLIME.

Usage

```
baylime_explain_instance(explainer,
  data_row,
  predict_fn,
  label = 2,
  num_features = 5,
  num_samples = 5000,
  seed = NULL)
```

Arguments

explainer	A baylime_explainer object.
data_row	The instance to explain.
predict_fn	Prediction function.
label	Column of predict_fn's output to explain.
num_features	Number of features kept in the local model.
num_samples	Number of perturbed samples.
seed	Optional seed for reproducibility.

Value

A list with the same structure as lime_explain_instance(), plus local_std, coef_var, posterior_alpha and posterior_lambda.

baylime_explainer	Create a BayLIME explainer
-------------------	----------------------------

Description

Create a BayLIME explainer.

Usage

```
baylime_explainer(training_data,  
  feature_names = NULL,  
  kernel_width = NULL,  
  feature_selection = "auto",  
  sample_around_instance = FALSE,  
  model_regressor = c(  
    "Bay_info_prior",  
    "non_Bay",  
    "Bay_non_info_prior",  
    "BayesianRidge_inf_prior_fit_alpha"  
  ),  
  prior_alpha = 1,  
  prior_lambda = 1000,  
  random_state = NULL)
```

Arguments

training_data	Numeric training-data matrix.
feature_names	Optional character vector of feature names.
kernel_width	Optional kernel width.
feature_selection	Feature-selection method.
sample_around_instance	If TRUE, perturbations are centred on the instance.
model_regressor	One of "Bay_info_prior", "non_Bay", "Bay_non_info_prior" or "BayesianRidge_inf_prior_fit_alpha".
prior_alpha	Informative prior on the noise precision (used by "Bay_info_prior").
prior_lambda	Informative prior on the weight precision (used by "Bay_info_prior" and "BayesianRidge_inf_prior_fit_alpha").
random_state	Optional default seed.

Value

An object of class baylime_explainer.

calculate_distances	<i>Euclidean distance of every row to an instance</i>
---------------------	---

Description

Euclidean distance of every row to an instance.

Usage

```
calculate_distances(scaled_data,
  instance_scaled)
```

Arguments

scaled_data	Numeric matrix in standardised space.
instance_scaled	Standardised instance to measure distances from.

Value

A numeric vector of Euclidean distances.

compute_cov_metrics	<i>Coefficient-of-variation metrics across repeated runs</i>
---------------------	--

Description

Coefficient-of-variation metrics across repeated runs.

Usage

```
compute_cov_metrics(run_rankings,
  topk = NULL,
  eps = 1e-12,
  missing_value = 0)
```

Arguments

run_rankings	List of data frames (feature, beta), one per run.
topk	If not NULL, keep only the topk features with the largest median abs(beta).
eps	Small constant used to avoid division by zero.
missing_value	Value used for a feature absent from a given run.

Value

A list with cov_results (per feature) and the averages cov_unscaled, cov_scaled, cov_iqr, median_sd.

compute_csi	<i>Coefficient Stability Index (CSI)</i>
-------------	--

Description

Coefficient Stability Index (CSI).

Usage

```
compute_csi(confidence_intervals,
            index_verbose = FALSE)
```

Arguments

confidence_intervals	List of named confidence-interval lists, one element per repeated LIME run.
index_verbose	If TRUE, print the per-feature overlap percentage.

Value

A single numeric value between 0 and 100.

compute_icsi	<i>Interval-Based Coefficient Stability Index (ICSI)</i>
--------------	--

Description

Interval-Based Coefficient Stability Index (ICSI).

Usage

```
compute_icsi(B,
             train_ratio = 0.8,
             eps = 1e-12)
```

Arguments

B	Coefficient matrix (runs x features).
train_ratio	Share of runs used to build the reference interval.
eps	Small constant used to avoid division by zero.

Value

A named list of coverage results (ICSI_MAD, ICSI_rMAD, ICSI_IQR, ICSI_sMAD), each with coverage_per_feature and overall_coverage.

compute_mad_stdevs	<i>Robust standard deviation from repeated coefficients</i>
--------------------	---

Description

Robust standard deviation from repeated coefficients.

Usage

```
compute_mad_stdevs(beta_list)
```

Arguments

beta_list	Matrix of coefficients, one row per repeated explanation.
-----------	---

Value

A numeric vector, $1.4826 * \text{MAD}$ for each column.

compute_mrms	<i>Median Rank Matching Stability of features across repeated runs</i>
--------------	--

Description

Median Rank Matching Stability of features across repeated runs.

Usage

```
compute_mrms(run_rankings,
              k = 5)
```

Arguments

run_rankings	List of data frames (feature, beta), one per run.
k	Number of top features used for the explanation-based top-k score.

Details

Features are ranked by decreasing absolute coefficient. Tied absolute coefficients receive the average of the rank positions they occupy. A missing feature is not assigned an artificial rank; it contributes zero to the matching score for that run. MRMS-All averages the resulting binary matches over all features and runs.

Value

A list with per_feature, mrms_all and mrms_topk. Scores are percentages.

compute_pcsi	<i>PCSI stability indices</i>
--------------	-------------------------------

Description

PCSI stability indices.

Usage

```
compute_pcsi(cov)
```

Arguments

cov The list returned by compute_cov_metrics().

Value

A list with PCSI_MAD, PCSI_sMAD, PCSI_IQR and PCSI_MSD.

compute_vsi	<i>Variable Stability Index (VSI)</i>
-------------	---------------------------------------

Description

Variable Stability Index (VSI).

Usage

```
compute_vsi(confidence_intervals,
             index_verbose = FALSE)
```

Arguments

confidence_intervals List of named confidence-interval lists, one element per repeated LIME run.

index_verbose If TRUE, print the overall percentage.

Details

For a pair of runs with different numbers of selected variables, the overlap is normalised by the smaller selected set, avoiding dependence on the first run's feature count.

Value

A single numeric value between 0 and 100.

compute_wls_stdevs	<i>Standard deviations of weighted ridge coefficients</i>
--------------------	---

Description

Standard deviations of weighted ridge coefficients.

Usage

```
compute_wls_stdevs(x,
  y,
  weights,
  alpha)
```

Arguments

x	Numeric predictor matrix used by the local model.
y	Numeric response vector.
weights	Sample weights.
alpha	Ridge penalty used when fitting the local model.

Value

A numeric vector of approximate coefficient standard deviations.

default_kernel_width	<i>Default LIME kernel width for tabular data</i>
----------------------	---

Description

Default LIME kernel width for tabular data.

Usage

```
default_kernel_width(n_features)
```

Arguments

n_features	Number of features in the data set.
------------	-------------------------------------

Value

A single numeric value, $0.75 * \sqrt{n_features}$.

fit_gp	<i>Fit a 1-D Gaussian process (Matern 3/2 kernel)</i>
--------	---

Description

Fit a 1-D Gaussian process (Matern 3/2 kernel).

Usage

```
fit_gp(x,
      y,
      alpha = 1e-6)
```

Arguments

x	Numeric vector of training inputs (kernel widths already evaluated).
y	Numeric vector of observed values (the loss at each x).
alpha	Observation noise variance added to the kernel diagonal.

Value

An object of class `optilime_gp`.

fit_weighted_ridge	<i>Weighted ridge regression with an unpenalised intercept</i>
--------------------	--

Description

Weighted ridge regression with an unpenalised intercept.

Usage

```
fit_weighted_ridge(x,
                  y,
                  w,
                  alpha = 0)
```

Arguments

x	Numeric predictor matrix.
y	Numeric response vector.
w	Numeric vector of sample weights.
alpha	Ridge penalty; alpha = 0 is weighted least squares.

Value

A list with `intercept`, `coef` and `fitted` (fitted values).

`generate_perturbations`

Generate a perturbed neighbourhood in standardised feature space

Description

Generate a perturbed neighbourhood in standardised feature space.

Usage

```
generate_perturbations(instance_scaled,
                        num_samples = 5000,
                        sample_around_instance = TRUE)
```

Arguments

`instance_scaled` Standardised instance (numeric vector).

`num_samples` Number of perturbed samples to generate; row 1 is the instance itself.

`sample_around_instance` If TRUE, sample around the instance; otherwise around the origin (the training-data mean).

Value

A numeric matrix (`num_samples` x `length(instance_scaled)`) in standardised space.

`lime_check_stability` *Repeat an explanation and compute every stability metric*

Description

Repeat an explanation and compute every stability metric.

Usage

```
lime_check_stability(explainer,
                     data_row,
                     predict_fn,
                     label = 2,
                     num_features = 5,
                     num_samples = 5000,
                     n_calls = 10,
                     index_verbose = FALSE,
                     topk = 5,
```

```
train_ratio = 0.8,
seed = explainer$random_state,
explain_fn = lime_explain_instance)
```

Arguments

explainer	An explainer object (from lime_explainer(), baylime_explainer() or alime_explainer()).
data_row	The instance to explain.
predict_fn	Prediction function passed on to explain_fn.
label	Column of predict_fn's output to explain.
num_features	Number of features kept in each individual explanation.
num_samples	Number of perturbed samples per explanation.
n_calls	Number of repeated explanations.
index_verbose	Passed on to compute_csi() / compute_vsi().
topk	Number of top features used for the top-k metrics.
train_ratio	Share of runs used as the reference set for compute_icsi().
seed	Seed for the whole procedure; the global RNG state is restored afterwards.
explain_fn	Function used to explain one instance, e.g. lime_explain_instance, baylime_explain_instance or alime_explain_instance.

Value

A list with csi, vsi, mrms_all, mrms_topk, PCSI_MAD/PCSI_sMAD/PCSI_IQR/PCSI_MSD, interval_results (ICSI) and the raw run_rankings / confidence_intervals.

```
lime_confidence_intervals
```

Confidence intervals for a LIME-style explanation

Description

Confidence intervals for a LIME-style explanation.

Usage

```
lime_confidence_intervals(explanation)
```

Arguments

explanation	The list returned by lime_explain_instance() or baylime_explain_instance().
-------------	---

Details

For Bayesian BayLIME fits, intervals are built from the posterior coefficient covariance. For LIME, ALIME and non-Bayesian BayLIME fits, approximate weighted-ridge standard errors are used.

Value

A named list of c(lower, upper) intervals, or NULL if uncertainty cannot be estimated.

lime_explain_instance *Explain a single instance with LIME*

Description

Explain a single instance with LIME.

Usage

```
lime_explain_instance(explainer,
  data_row,
  predict_fn,
  label = 2,
  num_features = 5,
  num_samples = 5000,
  seed = NULL)
```

Arguments

explainer	A lime_explainer object created by lime_explainer().
data_row	The instance to explain, in the original feature space.
predict_fn	Prediction function; takes a matrix and returns a numeric vector or a matrix of class probabilities.
label	Column of predict_fn's output to explain, when it returns a matrix.
num_features	Number of features to keep in the local model.
num_samples	Number of perturbed samples to generate.
seed	Optional seed; if NULL, the global RNG stream continues (repeated calls give different neighbourhoods).

Value

A list describing the local explanation (coefficients, local_exp, score, local_pred, and the data needed for confidence intervals).

lime_explainer	Create a LIME explainer
----------------	-------------------------

Description

Create a LIME explainer.

Usage

```
lime_explainer(training_data,  
               feature_names = NULL,  
               kernel_width = NULL,  
               feature_selection = "auto",  
               sample_around_instance = FALSE,  
               penalty = 1,  
               random_state = NULL)
```

Arguments

training_data	Numeric training-data matrix used to centre and scale features.
feature_names	Optional character vector of feature names.
kernel_width	Optional kernel width; defaults to default_kernel_width().
feature_selection	Feature-selection method passed to select_features().
sample_around_instance	If TRUE, perturbations are centred on the instance.
penalty	Ridge penalty of the local model (0 = weighted least squares).
random_state	Optional default seed used by lime_check_stability().

Value

An object of class lime_explainer.

lime_kernel	LIME exponential kernel
-------------	-------------------------

Description

LIME exponential kernel.

Usage

```
lime_kernel(distances,  
            kernel_width)
```

Arguments

distances	Numeric vector of distances to the instance being explained.
kernel_width	Positive numeric kernel width.

Value

Numeric vector of proximity weights, $\sqrt{\exp(-\text{distances}^2 / \text{kernel_width}^2)}$.

optilime	<i>OptiLIME: choose the LIME kernel width by Bayesian optimisation</i>
----------	--

Description

OptiLIME: choose the LIME kernel width by Bayesian optimisation.

Usage

```
optilime(explainer,
  data_row,
  predict_fn,
  bounds = c(0.1, 3.0),
  n_iters = 30,
  n_pre_samples = 10,
  label = 2,
  num_features = 5,
  num_samples = 5000,
  max_r2 = 0.9,
  epsilon = NULL,
  seed = 42,
  verbose = FALSE)
```

Arguments

explainer	A lime_explainer object.
data_row	The instance to explain.
predict_fn	Prediction function.
bounds	c(lower, upper) search bounds for the kernel width.
n_iters	Number of Bayesian-optimisation iterations.
n_pre_samples	Number of random initial kernel widths.
label	Column of predict_fn's output to explain.
num_features	Number of features kept in the local model.
num_samples	Number of perturbed samples per explanation.
max_r2	Target local R-squared passed to optilime_score().

epsilon	Optional tolerance passed to optilime_score().
seed	Seed used for every evaluation, so kernel widths are compared on identical perturbations.
verbose	If TRUE, print progress.

Value

A list with best_width, best_loss, r2, explanation, explainer and optimization.

optilime_score	<i>OptiLIME objective (reflected local R-squared)</i>
----------------	---

Description

OptiLIME objective (reflected local R-squared).

Usage

```
optilime_score(explanation,
               max_r2 = 0.9,
               epsilon = NULL)
```

Arguments

explanation	The list returned by lime_explain_instance().
max_r2	Target local R-squared; scores above it are penalised symmetrically.
epsilon	Optional tolerance used to flag and penalise degenerate explanations.

Value

A single numeric value to be maximised by optilime().

predict.optilime_gp	<i>Predict method for optilime_gp objects</i>
---------------------	---

Description

Predict method for optilime_gp objects.

Usage

```
predict.optilime_gp(object,
                    newdata,
                    return_std = FALSE,
                    ...)
```

Arguments

object	An optilime_gp object returned by fit_gp().
newdata	Numeric vector of new points.
return_std	If TRUE, also return the predictive standard deviation.
...	Unused; included for S3 consistency.

Value

A numeric vector of means, or a list with mean and sd when return_std = TRUE.

select_features	<i>Select the features used by a LIME-style local model</i>
-----------------	---

Description

Select the features used by a LIME-style local model.

Usage

```
select_features(x,
  y,
  w,
  num_features,
  method = "auto")
```

Arguments

x	Numeric predictor matrix.
y	Numeric response vector.
w	Sample weights.
num_features	Number of features to select.
method	One of "auto", "forward_selection", "highest_weights" or "none".

Value

An integer vector of selected column indices of x.

`weighted_r2`*Weighted R-squared*

Description

Weighted R-squared.

Usage

```
weighted_r2(y,  
  fitted,  
  w)
```

Arguments

<code>y</code>	Observed response.
<code>fitted</code>	Fitted values from the local model.
<code>w</code>	Sample weights.

Value

A single numeric value between (typically) 0 and 1.

Index

* **models**

- [alime_explain_instance](#), 3
- [alime_explainer](#), 3
- [bayesian_optimisation](#), 4
- [bayesian_ridge_fit](#), 5
- [baylime_explain_instance](#), 6
- [baylime_explainer](#), 7
- [calculate_distances](#), 8
- [compute_cov_metrics](#), 8
- [compute_csi](#), 9
- [compute_icsi](#), 9
- [compute_mad_stdevs](#), 10
- [compute_mrms](#), 10
- [compute_pcsi](#), 11
- [compute_vsi](#), 11
- [compute_wls_stdevs](#), 12
- [default_kernel_width](#), 12
- [fit_gp](#), 13
- [fit_weighted_ridge](#), 13
- [generate_perturbations](#), 14
- [lime_check_stability](#), 14
- [lime_confidence_intervals](#), 15
- [lime_explain_instance](#), 16
- [lime_explainer](#), 17
- [lime_kernel](#), 17
- [optilime](#), 18
- [optilime_score](#), 19
- [predict.optilime_gp](#), 19
- [select_features](#), 20
- [weighted_r2](#), 21

* **package**

- [limestab-package](#), 2

[alime_explain_instance](#), 3
[alime_explainer](#), 3

[bayesian_optimisation](#), 4
[bayesian_ridge_fit](#), 5
[baylime_explain_instance](#), 6
[baylime_explainer](#), 7

[calculate_distances](#), 8
[compute_cov_metrics](#), 8
[compute_csi](#), 9
[compute_icsi](#), 9
[compute_mad_stdevs](#), 10
[compute_mrms](#), 10
[compute_pcsi](#), 11
[compute_vsi](#), 11
[compute_wls_stdevs](#), 12

[default_kernel_width](#), 12

[fit_gp](#), 13
[fit_weighted_ridge](#), 13

[generate_perturbations](#), 14

[lime_check_stability](#), 14
[lime_confidence_intervals](#), 15
[lime_explain_instance](#), 16
[lime_explainer](#), 17
[lime_kernel](#), 17
[limestab \(limestab-package\)](#), 2
[limestab-package](#), 2

[optilime](#), 18
[optilime_score](#), 19

[predict.optilime_gp](#), 19

[select_features](#), 20

[weighted_r2](#), 21