Package: kumquat (via r-universe)

June 22, 2026

Title A Smaller Version of LIME

Version 1.1.0

Description The existing implementation of 'lime' can be quite limiting in understanding the underlying components that make Local Local interpretable model-agnostic explanations (LIME) work. 'kumquat' is a simpler implementation of 'lime' that is easier to understand and is more transparent on the pieces that come together to make LIME work. For more details on LIME, see Ribeiro, Singh, and Guestrin (2016) <doi:10.1145/2939672.2939778>.

License MIT + file LICENSE

Encoding UTF-8

Suggests colorspace, knitr, patchwork, randomForest, RColorBrewer, rmarkdown, testthat (>= 3.0.0), tidyverse

Config/testthat/edition 3

Imports bundle, dplyr, ggplot2, glmnet, glue, logger, rlang, tidyr

Depends R (>= 4.1.0)

LazyData true

VignetteBuilder knitr

URL https://janithwanni.github.io/kumquat/, https://github.com/janithwanni/kumquat

Config/roxygen2/version 8.0.0

BugReports https://github.com/janithwanni/kumquat/issues

NeedsCompilation no

Author Janith Wanniarachchi [aut, cre, cph]

Maintainer Janith Wanniarachchi <janithcwanni@gmail.com>

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

Date/Publication 2026-06-22 18:31:53 UTC

RemoteUrl https://github.com/cran/kumquat

RemoteRef HEAD

RemoteSha f1aeeeaa029f8c661cbb9fb6ba1154d86cdf0d57

Contents

`d_multi` *Complex Boundary Example Dataset*

Description

A demonstration dataset containing two numeric predictors and a binary class variable. The class boundary is set to A when x < (-0.5) & y < (-0.3) or when x >= -0.5 & x < 0.4 & y < 0.1 + 0.8 * x or when x >= 0.4 and B elsewhere

Usage

```
data(d_multi)
```

Format

A tibble with 5,000 rows and 3 variables:

x Numeric independent variable generated from a uniform distribution on [-1, 1]

y Numeric independent variable generated from a uniform distribution on [-1, 1]

class Binary categorical variable with two levels: "A" and "B"

Details

Simple two-dimensional dataset with a complex decison boundary.

d_multitwo *Combined Boundary Example Dataset*

Description

A demonstration dataset containing two numeric predictors and a binary class variable. The class boundary is set to A when x < (-0.5) & y < (-0.3) or when x >= -0.5 & x < 0.4 & y < 0.1 + 0.8 * x or when x >= 0.4 B when y < (-1) + 0.8 * x and everywhere else

Usage

```
data(d_multitwo)
```

Format

A tibble with 5,000 rows and 3 variables:

x Numeric independent variable generated from a uniform distribution on [-1, 1]

y Numeric independent variable generated from a uniform distribution on [-1, 1]

class Binary categorical variable with two levels: "A" and "B"

Details

Simple two-dimensional dataset with a combination of two decision boundaries

d_oblique *Oblique Boundary Example Dataset*

Description

A demonstration dataset containing two numeric predictors and a binary class variable. The class boundary is defined based on the inequality: y > 0.1 + 1.4 * x

Usage

```
data(d_oblique)
```

Format

A tibble with 5,000 rows and 3 variables:

x Numeric independent variable generated from a uniform distribution on [-1, 1]

y Numeric independent variable generated from a uniform distribution on [-1, 1]

class Binary categorical variable with two levels: "A" and "B"

Details

Simple two-dimensional dataset with an oblique decision boundary.

d_vertical *Vertical Boundary Example Dataset*

Description

A demonstration dataset containing two numeric predictors and a binary class variable. The class boundary is defined by a vertical split at x = 0.3.

Usage

```
data(d_vertical)
```

Format

A tibble with 5,000 rows and 3 variables:

x Numeric independent variable generated from a uniform distribution on [-1, 1]

y Numeric independent variable generated from a uniform distribution on [-1, 1]

class Binary categorical variable with two levels: "A" and "B"

Details

Simple two-dimensional dataset with a vertical decision boundary.

fit_local_model *Fit Local Linear Model and Compute Feature Importance*

Description

Fit Local Linear Model and Compute Feature Importance

Usage

```
fit_local_model(
  perturbations,
  predictor_vars,
  nfolds = 50,
  alpha = 1,
  class_names = c("A", "B")
)
```

Arguments

perturbations	Data frame of perturbations with predictions
predictor_vars	Character vector of predictor variable names
nfolds	Number of folds for cross-validation (default: 50)
alpha	Elastic net mixing parameter (default: 1 for lasso)
class_names	Character vector of class names for binary classification

Value

A list containing glm_predictions, importances, and the fitted model

Examples

```
perturbations <- data.frame(
  x1 = c(1, 2, 3),
  x2 = c(4, 5, 6),
  pred = c("A", "A", "A")
)

result <- fit_local_model(
  perturbations,
  predictor_vars = c("x1", "x2")
)
```

---

generate_perturbations

Generate Perturbations Around a Point of Interest

Description

Generate Perturbations Around a Point of Interest

Usage

```
generate_perturbations(
  data,
  poi,
  radius = 0.1,
  step = 0.01,
  predictors = names(data)
)
```

Arguments

`data`	Training data frame
`poi`	Row number of point of interest
`radius`	Perturbation radius (default: 0.1)
`step`	Step size for perturbations (default: 0.01)
`predictors`	Character vector of predictor variable names

Value

A data frame of perturbed points. The output contains the following properties:

- Columns are predictors
- Number of rows are going to be dependent on `radius` and `step`

Examples

```
data <- data.frame(
  x = 1,
  y = 2
)
result <- generate_perturbations(
  data,
  poi = 1,
  radius = 0.1,
  step = 0.1,
  predictors = c("x", "y")
 )
```

`kumquat` *Complete Local Interpretation Pipeline*

Description

Complete Local Interpretation Pipeline

Usage

```
kumquat(
  model_bundle,
  data,
  pois,
  perturbations = NULL,
  radius = 0.1,
  step = 0.01,
  predictor_vars = c("x", "y"),
  nfolds = 50,
  alpha = 1,
```

```
  class_names = c("A", "B"),
  predict_func = stats::predict
)
```

Arguments

model_bundle	A trained model held in a bundle::bundle()
data	Training data
pois	Points of interest (row numbers)
perturbations	A data.frame of perturbations to be used to fit the local model
radius	Perturbation radius (default: 0.1)
step	Perturbation step size (default: 0.01)
predictor_vars	Character vector of predictor variable names
nfolds	Number of CV folds (default: 50)
alpha	Elastic net parameter (default: 1)
class_names	Character vector of class names
predict_func	A function that takes in two arguments: model and data and returns a vector of factors

Value

A list containing perturbations, predictions, and local model results

Examples

```
data(d_vertical)
rfmodel <- randomForest::randomForest(
 class ~ x + y,
 data = d_vertical
)
# Bundle model up
rfmodel_bundled <- bundle::bundle(rfmodel)
ks <- kumquat(
 rfmodel_bundled,
 d_vertical,
  1,
  class_names = unique(d_vertical$class)
)
```

pinch_importance *Extract and Combine Feature Importances from Kumquat Objects*

Description

pinch_importance() extracts feature importance values from kumquats and combines them into a single data frame.

Usage

```
pinch_importance(kumquats)
```

Arguments

kumquats	A result from a call to make_kumquats

Value

A data.frame where each row corresponds to the feature importances extracted from one kumquat object. Column names represent feature names.

Examples

```
data(d_vertical)
rfmodel <- randomForest::randomForest(
 class ~ x + y,
 data = d_vertical
)
# Bundle model up
rfmodel_bundled <- bundle::bundle(rfmodel)
ks <- kumquat(
 rfmodel_bundled,
 d_vertical,
  1,
  class_names = unique(d_vertical$class)
)
imps <- pinch_importance(ks)
```

plot_interest *Plot point of interest with perturbations and predictions*

Description

This function generates a ggplot visualizing the perturbations of a local model, overlaying the training data and highlighting a specific point of interest (poi). The plot subtitle shows the importances of the first two features.

Usage

```
plot_interest(kquat)
```

Arguments

kquat	A list-like object containing at least: • local_model$importances numeric vector • local_model$glm_predictions numeric vector • perturbations data.frame with columns x and y • train_data data.frame with columns x, y, and class • poi integer or index of the point of interest

Value

A ggplot object

Examples

```
data(d_vertical)
rfmodel <- randomForest::randomForest(
 class ~ x + y,
 data = d_vertical
)
# Bundle model up
rfmodel_bundled <- bundle::bundle(rfmodel)
ks <- kumquat(
 rfmodel_bundled,
 d_vertical,
  1,
  class_names = unique(d_vertical$class)
)
plot_obj <- plot_interest(ks)
```

Index