Package: MLmorph (via r-universe)

June 7, 2026

Type Package

Title Integrating Morphological Modeling and Machine Learning for Decision Support

Version 0.1.1

Description Integrating morphological modeling with machine learning to support structured decision-making (e.g., in management and consulting). The package enumerates a morphospace of feasible configurations and uses random forests to estimate class probabilities over that space, bridging deductive model exploration with empirical validation. It includes utilities for factorizing inputs, model training, morphospace construction, and an interactive 'shiny' app for scenario exploration.

License MIT + file LICENSE

URL https://github.com/theogrost/MLmorph

BugReports https://github.com/theogrost/MLmorph/issues

Encoding UTF-8

LazyData true

Depends R (>= 4.3.0)

Imports bslib (>= 0.9.0), caret (>= 6.0.94), dplyr (>= 1.1.4), ggplot2 (>= 3.5.1), htmltools (>= 0.5.8.1), jsonlite (>= 1.8.8), magrittr, openxlsx (>= 4.2.5.2), plotly (>= 4.10.4), randomForest (>= 4.7.1.1), reactable (>= 0.4.4), shiny (>= 1.10.0), shinyFiles (>= 0.9.3), shinyjs (>= 2.1.0), stats (>= 4.3.0), tidyr (>= 1.3.1), utils (>= 4.3.0)

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Language en-US

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libssl-dev zlib1g-dev

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

Date/Publication 2025-11-09 20:08:29 UTC

RemoteUrl https://github.com/theogrost/mlmorph

RemoteRef HEAD

RemoteSha 48c1bbf86edb83a77d4f7e27fbcd02ecdbbebc5f

Contents

`create_morphospace` *Create a morphospace of predictor combinations with class probabilities*

Description

Create a morphospace of predictor combinations with class probabilities

Usage

```
create_morphospace(the_data, model, shiny = FALSE)
```

Arguments

`the_data`	A data.frame used to derive unique values of predictors.
`model`	A classification model fitted via a formula interface that supports `predict(model, newdata, type = "prob")` (e.g., from create_rf_model).
`shiny`	Logical; if `TRUE`, compute predictions in batches with **shiny** progress. Default `FALSE`.

Value

A list with components:

- `morphospace`: data frame with all predictor combinations, class label column (named as the dependent), `calculated` (probability), and `purely_simulated` flag.
- `dependent`: character scalar with the outcome name.
- `independent`: character vector of predictor names.
- `all_vars`: character vector `c(independent, dependent)`.
- `purely_simulated`: logical vector aligned with `morphospace`.

Examples

```
n  <- 60
y  <- factor(sample(letters[1:3], n, TRUE))
x1 <- factorize_numeric_vector(runif(n, 10, 20), breaks_no = 3)
x2 <- factorize_numeric_vector(runif(n,  1,  2), breaks_no = 3)
df <- data.frame(y, x1, x2)
fit <- create_rf_model(df, dependent = "y", ntree = 50)$model
ms  <- create_morphospace(df, fit)
names(ms)
```

`create_rf_model` *Create a random forest classification model*

Description

Create a random forest classification model

Usage

```
create_rf_model(
  data,
  dependent = colnames(data)[ncol(data)],
  independent = setdiff(colnames(data), dependent),
  train_validate_split = 0.8,
  shiny = FALSE,
  ...
)
```

Arguments

`data`	A data.frame containing predictors and the outcome.
`dependent`	Character scalar; the name of the outcome (must be a factor for classification). Defaults to the last column of `data`.
`independent`	Character vector; names of predictor variables. Defaults to all columns except `dependent`.

`train_validate_split`
: Numeric in (0, 1); proportion of rows used for training. Default is `0.8`.

`shiny`
: Logical; if `TRUE`, trains incrementally and reports progress via MLmorph app. Default `FALSE`.

`...`
: Additional arguments passed to randomForest (e.g., `ntree`).

Value

A named list with components:

- `model`: a randomForest return object.
- `variables_importance`: matrix from importance.
- `model_performance_on_test`: a confusionMatrix return object on the validation set.

Examples

```
n <- 60
y  <- factor(sample(letters[1:3], n, TRUE))
x1 <- factorize_numeric_vector(runif(n, 10, 20), breaks_no = 3)
x2 <- factorize_numeric_vector(runif(n,  1,  2), breaks_no = 5)
df <- data.frame(y, x1, x2)
fit <- create_rf_model(df, dependent = "y", ntree = 50)
names(fit)
```

---

factorize_binary_vector

Turn binary vector into a factor

Description

Turn binary vector into a factor

Usage

```
factorize_binary_vector(data_vector, custom_labels = NULL)
```

Arguments

`data_vector`
: Logical vector.

`custom_labels`
: Optional length-2 character vector: first for `TRUE`, second for `FALSE`.

Value

A factor with two levels in `TRUE`, `FALSE` order.

Examples

```
factorize_binary_vector(c(TRUE, FALSE, TRUE))
```

factorize_character_vector *Turn character vector into a factor*

Description

Turn character vector into a factor

Usage

```
factorize_character_vector(data_vector, custom_labels = NULL)
```

Arguments

data_vector	Character vector.
custom_labels	Optional named character vector where names are original values and values are labels.

Value

A factor with labeled levels.

Examples

```
factorize_character_vector(c("A First", "B Second", "C Third"))
```

factorize_identity *Identity factorization for numbered strings*

Description

Identity factorization for numbered strings

Usage

```
factorize_identity(data_vector)
```

Arguments

data_vector	Character vector where values are already labeled (e.g., "1. A").

Value

A factor with `levels == labels`.

Examples

```
factorize_identity(c("1. First", "2. Second", "3. Third"))
```

factorize_nicely_dataframe

Heuristic factorization for all columns of a data frame

Description

Heuristic factorization for all columns of a data frame

Usage

```
factorize_nicely_dataframe(data_frame)
```

Arguments

data_frame	A data frame.

Value

A data frame with all columns converted to factors.

Examples

```
df <- data.frame(x = runif(20), y = rep(c(TRUE, FALSE, TRUE, TRUE), 5))
factorize_nicely_dataframe(df)
```

factorize_nicely_vector

Heuristic factorization for a single vector

Description

Heuristic factorization for a single vector

Usage

```
factorize_nicely_vector(data_vector)
```

Arguments

data_vector	A vector (numeric, logical, or character).

Value

A factor (ordered for numeric inputs with many distinct values).

Examples

```
factorize_nicely_vector(c("a", "b", "a"))
```

`factorize_numeric_vector` *Turn numeric vector into an ordered factor*

Description

Turn numeric vector into an ordered factor

Usage

```
factorize_numeric_vector(
  data_vector,
  method = c("equal_bins", "equal_distance", "custom_breaks"),
  breaks_no = 5,
  custom_breaks = NULL,
  custom_labels = NULL
)
```

Arguments

`data_vector`	Numeric vector.
`method`	Factorization rule: one of `"equal_bins"`, `"equal_distance"`, `"custom_breaks"`.
`breaks_no`	Integer ≥ 2; number of intervals when `method != "custom_breaks"`.
`custom_breaks`	Optional numeric vector of cut points (strictly increasing) used when `method = "custom_breaks"`.
`custom_labels`	Optional character vector of labels. If supplied, its length should equal `length(custom_breaks) - 1`.

Value

An ordered factor with interval labels.

Examples

```
factorize_numeric_vector(runif(10))
```

load_data *Load tabular data (xlsx, csv, or json)*

Description

Load tabular data (xlsx, csv, or json)

Usage

```
load_data(data_path)
```

Arguments

data_path	Character scalar; path to a .xlsx, .csv, or .json file.

Value

A base data.frame with the imported data.

Examples

```
tmp_csv <- tempfile(fileext = ".csv")
utils::write.csv(data.frame(a = 1:2, b = c("x", "y")), tmp_csv, row.names = FALSE)
load_data(tmp_csv)

tmp_json <- tempfile(fileext = ".json")
jsonlite::write_json(list(a = 1:2, b = c("x","y")), tmp_json, auto_unbox = TRUE)
load_data(tmp_json)

tmp_xlsx <- tempfile(fileext = ".xlsx")
openxlsx::write.xlsx(data.frame(a = 1:2, b = c("x","y")), tmp_xlsx)
load_data(tmp_xlsx)
```

MLmorph *Launch the MLmorph shiny app*

Description

Launch the MLmorph shiny app

Usage

```
MLmorph(
  host = "127.0.0.1",
  port = NULL,
  launch.browser = TRUE,
  maxUploadSize = 200 * 1024^2
)
```

Arguments

`host`	Host interface to bind (default `"127.0.0.1"`).
`port`	Integer port or `NULL` to auto-select.
`launch.browser`	Logical; open in a browser. Default `TRUE`.
`maxUploadSize`	Maximum request size in bytes; sets `options(shiny.maxRequestSize = ...)`. Default `200 * 1024^2`.

Value

The value returned by runApp.

See Also

`runApp`

Examples

```
if(interactive()){
MLmorph()
}
```

Index