Package ‘AutoNLS’

November 25, 2024

Type Package

Title Automated Non-Linear Regression

Version 0.1.0

Author Adrian Antico [aut, cre, cph]

Maintainer Adrian Antico <adrianantico@gmail.com>

Description AutoNLS is a comprehensive package for automated non-linear regression modeling, evaluation, and visualization. It supports dynamic selection of non-linear models, tools for scoring and comparing models, and powerful visualizations using the `echarts4r` package. The package is designed for ease of use and extensibility, making it ideal for analysts, data scientists, and researchers.

Imports R6,
data.table,
dplyr,
echarts4r,
minpack.lm,
mgcv

Suggests testthat

License AGPL (>= 3)

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

URL https://github.com/AdrianAntico/AutoNLS

BugReports https://github.com/AdrianAntico/AutoNLS/issues

Language en-US

NeedsCompilation no

R topics documented:

EDA *EDA (Exploratory Data Analysis) Class*

Description

Provides tools for automated exploratory data analysis, including summary statistics, correlation matrices, and customizable visualizations using `echarts4r`.

Methods

- `initialize(data)`: Initializes the class with a `data.table`.
- `summarize()`: Computes summary statistics.
- `correlate()`: Computes a correlation matrix for numeric columns.
- `visualize_distributions()`: Creates histogram and density visualizations for numeric columns.
- `visualize_scatterplots()`: Creates pairwise scatterplots for numeric columns.
- `render_all()`: Runs all methods and returns their results.

Public fields

`data` A `data.table` containing the dataset for analysis.

`summary_stats` A `data.table` storing the summary statistics of the dataset.

`correlation_matrix` A correlation matrix for numeric columns.

`plots` A list of `echarts4r` plots generated during the analysis. Initialize the EDA class

Methods

Public methods:

- `EDA$new()`
- `EDA$summarize()`
- `EDA$correlate()`
- `EDA$visualize_distributions()`
- `EDA$visualize_scatterplots()`
- `EDA$render_all()`
- `EDA$clone()`

Method `new()`:

Usage:

```
EDA$new(data)
```

Arguments:

`data` A `data.table` containing the dataset for analysis. Compute summary statistics
Calculates mean, median, variance, and the count of missing values for each column.

Method `summarize()`:

Usage:

```
EDA$summarize()
```

Returns: A `data.table` containing the summary statistics. Compute correlation with the target variable
Calculates both Pearson and Spearman correlations between all numeric columns (excluding the target variable) and the target variable.

Method `correlate()`:

Usage:

```
EDA$correlate(target_col = "y")
```

Arguments:

`target_col` the target variable in the data set

Returns: A data.table with the Pearson and Spearman correlation values for each numeric predictor. Visualize distributions with histograms and optional density lines
Generates histograms for numeric columns and optionally overlays density lines.

Method `visualize_distributions()`:

Usage:

```
EDA$visualize_distributions(
  title_prefix = "Distribution of",
  bins = NULL,
  add_density = TRUE,
  density_color = "#EE6666",
  tooltip_trigger = "axis",
  theme = "dark",
  density_opacity = 0.4
)
```

Arguments:

`title_prefix` Character. Prefix for the plot title.

`bins` Integer. Number of bins for the histogram. Defaults to Sturges' formula.

`add_density` Logical. Whether to add a density line. Defaults to `TRUE`.

`density_color` Character. Color for the density line. Defaults to `"#EE6666"`.

`tooltip_trigger` "axis"

`theme` Character. Theme for the plot (e.g., "light", "dark"). Defaults to `"light"`.

`density_opacity` numeric. default 0.4

Returns: A list of `echarts4r` histogram plots. Visualize pairwise scatterplots with GAM fits
Generates scatterplots for all pairs of numeric columns and overlays fitted lines from Generalized Additive Models (GAM) for different `k` values.

Method `visualize_scatterplots()`:

Usage:

```
EDA$visualize_scatterplots(
  title_prefix = "Scatterplot of",
  theme = "dark",
  k_values = c(3, 5, 7)
)
```

Arguments:

title_prefix Character. Prefix for the plot title.

theme Character. Theme for the plot (e.g., "light", "dark"). Defaults to "light".

k_values Numeric vector. Values of k (basis dimension) for GAM fits. Defaults to c(3, 5, 7).

Returns: A list of echarts4r scatter plots with GAM fitted lines. Render All Visualizations This method generates all visualizations, including distributions and scatterplots.

Method render_all()**:**

Usage:

```
EDA$render_all(
  y_col = NULL,
  dist_title_prefix = "Distribution of",
  dist_bins = 10,
  dist_add_density = TRUE,
  dist_density_color = "#EE6666",
  dist_theme = "light",
  scatter_title_prefix = "Scatterplot of"
)
```

Arguments:

dist_title_prefix Prefix for titles of distribution plots.

dist_bins Number of bins for histograms in distribution plots.

dist_add_density Logical. Whether to overlay a density line on histograms.

dist_density_color Color for the density line.

dist_theme Visualization theme for the distribution plots.

scatter_title_prefix Prefix for titles of scatterplot visualizations.

Returns: A list of generated plots.

Method clone()**:** The objects of this class are cloneable with this method.

Usage:

```
EDA$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

ModelVisualizer *ModelVisualizer*

Description

An R6 class to visualize the shapes of various non-linear models for comparison.

Public fields

models A list of non-linear models and their parameterized functions. Initialize the ModelVisualizer class

Methods

Public methods:

- `ModelVisualizer$new()`
- `ModelVisualizer$generate_comparison_plot()`
- `ModelVisualizer$clone()`

Method `new()`**:**

Usage:

```
ModelVisualizer$new(models)
```

Arguments:

`models` A list of models with parameterized functions.

Returns: A new instance of the ModelVisualizer class.

Method `generate_comparison_plot()`**:**

Usage:

```
ModelVisualizer$generate_comparison_plot(
  x_range = seq(1, 100, by = 1),
  params = list(),
  normalize = TRUE
)
```

Arguments:

`x_range` A numeric vector specifying the range of x values to evaluate (e.g., `seq(1, 100, by = 1)`).

`params` A named list of parameters for each model. Defaults to an empty list, which uses default parameters for all models.

`normalize` Logical. If TRUE, normalizes the y values for each model to fall between 0 and 1. Defaults to TRUE.

Returns: An `echarts4r` object representing the comparison plot.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ModelVisualizer$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`NonLinearFitter` *NonLinearFitter*

Description

An R6 class for automatically fitting non-linear regression models. Includes a library of pre-defined models to simplify selection.

Public fields

`data` A data.table containing the dataset for modeling.

`models` A list of non-linear models to test.

`fit_results` A list to store the results of model fits.

`evaluation_metrics` A list to store evaluation metrics for each model.

`plots` A list to store plots of model fits.

`model_library` A pre-defined library of common non-linear models. Initialize the NonLinearFitter class

Methods

Public methods:

- `NonLinearFitter$new()`
- `NonLinearFitter$list_models()`
- `NonLinearFitter$add_model()`
- `NonLinearFitter$fit_models()`
- `NonLinearFitter$clone()`

Method `new()`**:**

Usage:

```
NonLinearFitter$new(data)
```

Arguments:

`data` A data.table containing the dataset for modeling. Must include the predictor and response variable columns.

Returns: A new instance of the NonLinearFitter class.

Examples:

```
data <- data.table::data.table(x = 1:100, y = 5 / (1 + exp(-0.1 * (1:100 - 50))))
fitter <- NonLinearFitter$new(data)
```

Method `list_models()`**:**

Usage:

```
NonLinearFitter$list_models()
```

Returns: A data.table summarizing available models. Add a non-linear model for testing

Method `add_model()`**:**

Usage:

```
NonLinearFitter$add_model(name, formula = NULL, start_params = NULL)
```

Arguments:

`name` The name of the model (e.g., "Hill").

`formula` The non-linear formula for the model (optional if using pre-defined model).

`start_params` A list of starting parameters for the model (optional if using pre-defined model).

Returns: NULL

Examples:

```
# Add a pre-defined model
fitter$add_model("Hill")

# Add a custom model
fitter$add_model("Custom", y ~ a * exp(-b * x), list(a = 1, b = 0.1))
```

Method `fit_models()`**:**

Usage:

```
NonLinearFitter$fit_models(x_col, y_col, control = list(maxiter = 200))
```

Arguments:

`x_col` The name of the predictor variable.

`y_col` The name of the response variable.

`control` A list of control parameters for the optimizer, such as `maxiter`. Default is `list(maxiter = 200)`.

Returns: A list of fitted model objects.

Examples:

```
fitter$fit_models(x_col = "x", y_col = "y", control = list(maxiter = 200))
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
NonLinearFitter$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## ------------------------------------------------
## Method `NonLinearFitter$new`
## ------------------------------------------------

data <- data.table::data.table(x = 1:100, y = 5 / (1 + exp(-0.1 * (1:100 - 50))))
fitter <- NonLinearFitter$new(data)

## ------------------------------------------------
## Method `NonLinearFitter$add_model`
## ------------------------------------------------

# Add a pre-defined model
fitter$add_model("Hill")

# Add a custom model
fitter$add_model("Custom", y ~ a * exp(-b * x), list(a = 1, b = 0.1))

## ------------------------------------------------
## Method `NonLinearFitter$fit_models`
## ------------------------------------------------

fitter$fit_models(x_col = "x", y_col = "y", control = list(maxiter = 200))
```

`NonLinearModelEvaluator` *NonLinearModelEvaluator*

Description

An R6 class to evaluate non-linear regression models. Includes tools to generate tables of statistics and visualizations to compare models against data.

Public fields

`fit_results` A list of fitted model objects.

`evaluation_metrics` A data.table containing model evaluation metrics.

`plots` A list of visualizations comparing models against data.

`data` The original dataset used for fitting models. Initialize the NonLinearModelEvaluator class

Methods

Public methods:

- `NonLinearModelEvaluator$new()`
- `NonLinearModelEvaluator$generate_metrics()`
- `NonLinearModelEvaluator$generate_comparison_plot()`
- `NonLinearModelEvaluator$clone()`

Method `new()`**:**

Usage:

```
NonLinearModelEvaluator$new(fit_results, data)
```

Arguments:

`fit_results` A list of fitted model objects (e.g., output from NonLinearFitter).

`data` The original dataset used for fitting models.

Returns: A new instance of the NonLinearModelEvaluator class.

Method `generate_metrics()`**:**

Usage:

```
NonLinearModelEvaluator$generate_metrics()
```

Returns: A data.table of evaluation metrics.

Method `generate_comparison_plot()`**:**

Usage:

```
NonLinearModelEvaluator$generate_comparison_plot(data, x_col, y_col)
```

Arguments:

`data` A `data.table` or `data.frame` containing the dataset used for evaluation.

`x_col` A string specifying the name of the x variable in the dataset.

`y_col` A string specifying the name of the y variable in the dataset.

Returns: An `echarts4r` plot showing observed vs. predicted data.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
NonLinearModelEvaluator$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`NonLinearModelScorer` *NonLinearModelScorer*

Description

An R6 class to score non-linear regression models on new data and visualize the results.

Public fields

`fit_results` A list of fitted model objects.

`scored_data` A list of data.tables containing scored data.

`score_plots` A list of plots visualizing scored data.

Methods

Public methods:

- `NonLinearModelScorer$new()`
- `NonLinearModelScorer$score_new_data()`
- `NonLinearModelScorer$generate_score_plot()`
- `NonLinearModelScorer$clone()`

Method `new()`**:**

Usage:

```
NonLinearModelScorer$new(fit_results)
```

Arguments:

`fit_results` A list of fitted model objects (e.g., output from NonLinearFitter).

Returns: A new instance of the NonLinearModelScorer class.

Method `score_new_data()`**:**

Usage:

```
NonLinearModelScorer$score_new_data(new_data, x_col)
```

Arguments:

`new_data` A data.table containing the new data to score.

`x_col` The predictor column in `new_data`.

Returns: A list of data.tables with predicted values for each model.

Method `generate_score_plot()`**:**

Usage:

```
NonLinearModelScorer$generate_score_plot(model_name, new_data, x_col)
```

Arguments:

`model_name` The name of the model to plot.

`new_data` The original new data used for scoring.

`x_col` The predictor column in `new_data`.

Returns: A plot visualizing the scored data.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
NonLinearModelScorer$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Index