

Package ‘tidypredict’

November 29, 2025

Title Run Predictions Inside the Database

Version 1.0.0

Description It parses a fitted 'R' model object, and returns a formula in 'Tidy Eval' code that calculates the predictions. It works with several databases back-ends because it leverages 'dplyr' and 'dbplyr' for the final 'SQL' translation of the algorithm. It currently supports `lm()`, `glm()`, `randomForest()`, `ranger()`, `earth()`, `xgb.Booster.complete()`, `cubist()`, and `ctree()` models.

License MIT + file LICENSE

URL <https://tidypredict.tidymodels.org>,
<https://github.com/tidymodels/tidypredict>

BugReports <https://github.com/tidymodels/tidypredict/issues>

Depends R (>= 3.6)

Imports cli, dplyr (>= 0.7), generics, knitr, purrr, rlang (>= 1.1.1),
tibble, tidyr

Suggests covr, Cubist (>= 0.5.1), DBI, dbplyr, earth (>= 5.1.2),
glmnet, methods, mlbench, modeldata, nycflights13, parsnip,
partykit, randomForest, ranger, rmarkdown, RSQLite, testthat
(>= 3.2.0), xgboost, yaml

VignetteBuilder knitr

Config/Needs/website tidyverse/tidytemplate

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Emil Hvitfeldt [aut, cre],
Edgar Ruiz [aut],
Max Kuhn [aut]

Maintainer Emil Hvitfeldt <emil.hvitfeldt@posit.co>

Repository CRAN

Date/Publication 2025-11-29 06:10:02 UTC

Contents

acceptable_formula	2
as_parsed_model	3
generate_tree_node	3
parse_model	4
path_formula	4
path_formulas	5
tidy.pm_regression	5
tidypredict_fit	6
tidypredict_interval	6
tidypredict_test	7
tidypredict_to_column	8
Index	9

acceptable_formula	<i>Checks that the formula can be parsed</i>
--------------------	--

Description

Uses an S3 method to check that a given formula can be parsed based on its class. It currently scans for contrasts that are not supported and in-line functions. (e.g: `lm(wt ~ as.factor(am))`). Since this function is meant for function interaction, as opposed to human interaction, a successful check is silent.

Usage

```
acceptable_formula(model)
```

Arguments

model An R model object

Examples

```
model <- lm(mpg ~ wt, mtcars)
acceptable_formula(model)
```

as_parsed_model	<i>Prepares parsed model object</i>
-----------------	-------------------------------------

Description

Prepares parsed model object

Usage

```
as_parsed_model(x)
```

Arguments

x	A parsed model object
---	-----------------------

generate_tree_node	<i>Construct a single node of a tree</i>
--------------------	--

Description

Construct a single node of a tree

Usage

```
generate_tree_node(node, calc_mode = "")
```

Arguments

node	a list with named elements path and prediction. See details for more.
calc_mode	character, takes values "" and "calc_mode". The node list should contain the two lists path and prediction. The path element has the following structure: This list can contain 0 or more elements. The elements but each be of the following format: • type character, must be "conditional", "set", or "all". • op character. if type == "conditional" must be "more", "more-equal", "less", or "less-equal". if type == "set" must be "in" or not-in. • col character. • val if type == "conditional" and vals if type == "set". Can be character or numeric. The prediction list has the following structure: It can either be a singular value or a list. If it is a list it will have the following 4 named elements col, val, op, and is_intercept.

- col character, name of column
 - val val, numeric of character
 - op character, known values are "none" and "multiply". "none" is used then is_intercept == 1.
 - is_interceptinteger, takes values 0 and 1.'
- @keywords internal

parse_model	<i>Converts an R model object into a table.</i>
-------------	---

Description

It parses a fitted R model’s structure and extracts the components needed to create a dplyr formula for prediction. The function also creates a data frame using a specific format so that other functions in the future can also pass parsed tables to a given formula creating function.

Usage

```
parse_model(model)
```

Arguments

model An R model object.

Examples

```
library(dplyr)
df <- mutate(mtcars, cyl = paste0("cyl", cyl))
model <- lm(mpg ~ wt + cyl * disp, offset = am, data = df)
parse_model(model)
```

path_formula	<i>Turn a path object into an expression</i>
--------------	--

Description

Turn a path object into an expression

Usage

```
path_formula(x)
```

Arguments

- `x` a list.
- The input of this function is a list with 4 values.
- type character, must be "conditional" or "set".
 - op character. if type == "conditional" must be "more", "more-equal", "less", or "less-equal". if type == "set" must be "in" or "not-in".
 - col character.
 - val if type == "conditional" and vals if type == "set". Can be character or numeric. @keywords internal

path_formulas	<i>Turn a path object into a combined expression</i>
---------------	--

Description

Turn a path object into a combined expression

Usage

```
path_formulas(path)
```

Arguments

- `path` a list of lists.
- This list can contain 0 or more elements. The elements but each be of the following format:
- type character, must be "conditional", "set", or "all".
 - op character. if type == "conditional" must be "more", "more-equal", "less", or "less-equal". if type == "set" must be "in" or "not-in".
 - col character.
 - val if type == "conditional" and vals if type == "set". Can be character or numeric. @keywords internal

tidy.pm_regression	<i>Tidy the parsed model results</i>
--------------------	--------------------------------------

Description

Tidy the parsed model results

Usage

```
## S3 method for class 'pm_regression'
tidy(x, ...)
```

Arguments

x	A parsed_model object
...	Reserved for future use

tidypredict_fit	<i>Returns a Tidy Eval formula to calculate fitted values</i>
-----------------	---

Description

It parses a model or uses an already parsed model to return a Tidy Eval formula that can then be used inside a dplyr command.

Usage

```
tidypredict_fit(model)
```

Arguments

model	An R model or a list with a parsed model.
-------	---

Examples

```
model <- lm(mpg ~ wt + cyl * disp, offset = am, data = mtcars)
tidypredict_fit(model)
```

tidypredict_interval	<i>Returns a Tidy Eval formula to calculate prediction interval.</i>
----------------------	--

Description

It parses a model or uses an already parsed model to return a Tidy Eval formula that can then be used inside a dplyr command.

Usage

```
tidypredict_interval(model, interval = 0.95)
```

Arguments

model	An R model or a list with a parsed model
interval	The prediction interval, defaults to 0.95

Details

The result still has to be added to and subtracted from the fit to obtain the upper and lower bound respectively.

Examples

```
model <- lm(mpg ~ wt + cyl * disp, offset = am, data = mtcars)
tidypredict_interval(model)
```

tidypredict_test	<i>Tests base predict function against tidypredict</i>
------------------	--

Description

Compares the results of predict() and tidypredict_to_column() functions.

Usage

```
tidypredict_test(
  model,
  df = model$model,
  threshold = 1e-12,
  include_intervals = FALSE,
  max_rows = NULL,
  xg_df = NULL
)
```

Arguments

model	An R model or a list with a parsed model. It currently supports lm(), glm() and randomForest() models.
df	A data frame that contains all of the needed fields to run the prediction. It defaults to the "model" data frame object inside the model object.
threshold	The number that a given result difference, between predict() and tidypredict_to_column() should not exceed. For continuous predictions, the default value is 0.000000000001 (1e-12), and for categorical predictions, the default value is 0.
include_intervals	Switch to indicate if the prediction intervals should be included in the test. It defaults to FALSE.
max_rows	The number of rows in the object passed in the df argument. Highly recommended for large data sets.
xg_df	A xgb.DMatrix object, required only for XGBoost models. It defaults to NULL recommended for large data sets.

Examples

```
model <- lm(mpg ~ wt + cyl * disp, offset = am, data = mtcars)
tidypredict_test(model)
```

`tidypredict_to_column` *Adds the prediction columns to a piped command set.*

Description

Adds a new column with the results from `tidypredict_fit()` to a piped command set. If `add_interval` is set to `TRUE`, it will add two additional columns- one for the lower and another for the upper prediction interval bounds.

Usage

```
tidypredict_to_column(  
  df,  
  model,  
  add_interval = FALSE,  
  interval = 0.95,  
  vars = c("fit", "upper", "lower")  
)
```

Arguments

<code>df</code>	A data.frame or tibble
<code>model</code>	An R model or a parsed model inside a data frame
<code>add_interval</code>	Switch that indicates if the prediction interval columns should be added. Defaults to <code>FALSE</code>
<code>interval</code>	The prediction interval, defaults to 0.95. Ignored if <code>add_interval</code> is set to <code>FALSE</code>
<code>vars</code>	The name of the variables that this function will produce. Defaults to "fit", "upper", and "lower".

Index

`acceptable_formula`, [2](#)
`as_parsed_model`, [3](#)

`generate_tree_node`, [3](#)

`parse_model`, [4](#)
`path_formula`, [4](#)
`path_formulas`, [5](#)

`tidy.pm_regression`, [5](#)
`tidypredict_fit`, [6](#)
`tidypredict_interval`, [6](#)
`tidypredict_test`, [7](#)
`tidypredict_to_column`, [8](#)