Package 'shinymodels'

November 17, 2021

Title Interactive Assessments of Models

Version 0.1.0

Description Launch a 'shiny' application for 'tidymodels' results. For classification or regression models, the app can be used to determine if there is lack of fit or poorly predicted points.

License MIT + file LICENSE

URL https://shinymodels.tidymodels.org, https://github.com/tidymodels/shinymodels

BugReports https://github.com/tidymodels/shinymodels/issues

Depends ggplot2, R (>= 2.10)

Imports dplyr, DT, generics (>= 0.1.0), glue, htmltools, magrittr, parsnip, plotly, purrr, rlang, scales, shiny, shinydashboard, stats, tidyr, tidyselect, tune, yardstick

Suggests covr, knitr, markdown, modeldata, rmarkdown, shinytest, spelling, testthat (>= 3.0.0)

Config/Needs/website tidyverse/tidytemplate

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData false

RoxygenNote 7.1.2

NeedsCompilation no

Author Max Kuhn [aut, cre] (<https://orcid.org/0000-0003-2402-136X>), Shisham Adhikari [aut], Julia Silge [aut] (<https://orcid.org/0000-0002-3671-836X>), RStudio [cph]

Maintainer Max Kuhn <max@rstudio.com>

Repository CRAN

Date/Publication 2021-11-17 21:00:02 UTC

R topics documented:

`ames_mlp_itr` *Iterative optimization of neural network*

Description

This object has the results when a neural network was tuned using Bayesian optimization and a validation set.

Details

The code used to produce this object:

```
data(ames)

ames <-
  ames %>%
  select(Sale_Price, Neighborhood, Longitude, Latitude, Year_Built) %>%
  mutate(Sale_Price = log10(ames$Sale_Price))

set.seed(1)
ames_rs <- validation_split(ames)

ames_rec <-
  recipe(Sale_Price ~ ., data = ames) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors()) %>%
  step_normalize(all_predictors())

mlp_spec <-
  mlp(hidden_units = tune(),
      penalty = tune(),
      epochs = tune()) %>%
  set_mode("regression")

set.seed(1)
ames_mlp_itr <-
  mlp_spec %>%
```

```
  tune_bayes(
    ames_rec,
    resamples = ames_rs,
    initial = 5,
    iter = 4,
    control = control_bayes(save_pred = TRUE)
  )
```

Value

An object with primary class `iteration_results`.

`cars_bag_vfld` *Resampled bagged tree results*

Description

This object has the results when a bagged regression tree was resampled using 10-fold cross-validation.

Details

The code used to produce this object:

```
library(tidymodels)
library(baguette)
tidymodels_prefer()

# ------------------------------------------------------------------------------

ctrl_rs <- control_resamples(save_pred = TRUE)

# ------------------------------------------------------------------------------

set.seed(1)
cars_rs <- vfold_cv(mtcars)

cars_bag_vfld <-
  bag_tree() %>%
  set_engine("rpart", times = 5) %>%
  set_mode("regression") %>%
  fit_resamples(
    mpg ~ .,
    resamples = cars_rs,
    control = ctrl_rs
  )
```

Value

An object with primary class `resample_results`.

`cell_race` *A CART classification tree tuned via racing*

Description

This object has the results when a CART classification tree model was tuned over the cost-complexity parameter using racing.

Details

To reduce the object size, a smaller subset of the data were used.

The code used to produce this object:

```
library(tidymodels)
library(finetune)
tidymodels_prefer()

ctrl_rc <- control_race(save_pred = TRUE)

# ------------------------------------------------------------------------------

data(cells)

set.seed(1)
cells <-
  cells %>%
  select(-case) %>%
  sample_n(200)

# ------------------------------------------------------------------------------

set.seed(2)
cell_rs <- vfold_cv(cells)

# ------------------------------------------------------------------------------

set.seed(3)
cell_race <-
  decision_tree(cost_complexity = tune()) %>%
  set_mode("classification") %>%
  tune_race_anova(
    class ~ .,
    resamples = cell_rs,
```

```
  grid = tibble(cost_complexity = 10^seq(-2, -1, by = 0.2)),
  control = ctrl_rc
)
```

Value

An object with primary class `tune_race`.

`explore.default` *Explore model results*

Description

`explore()` launches a Shiny application to interact with results from some tidymodels functions.

To investigate model fit(s), `explore()` can be used on objects produced by

- `tune::fit_resamples()`
- `tune::tune_grid()`
- `tune::tune_bayes()`
- `finetune::tune_sim_anneal()`
- `finetune::tune_race_anova()`
- `finetune::tune_race_win_loss()`
- `tune::last_fit()`

The application starts in a new window and allows users to see how predicted values align with the true, observed data. There are 2-3 tabs in the application (depending on the object):

- **Tuning Parameters** enables users to choose a specific set of tuning parameters. These results are shown in the **Plots** tab. The default configuration is based on the *optimal value* of the first performance metric used during the creation of the object.
- **Plots** shows various panels that can visualize how well the model fits. Specific points can be highlighted by clicking on them (as long as the `hover_only = FALSE` option was used). To reset the highlighted points, double on the graph background.
- **About** gives information on the application as well as links to get help or file bug reports/feature requests.

To quit the Shiny application, use the `Esc` key.

Usage

```
## Default S3 method:
explore(x, ...)

## S3 method for class 'tune_results'
explore(x, hover_cols = NULL, hover_only = FALSE, ...)
```

Arguments

x	An object with class tune_results.
...	Other parameters not currently used.
hover_cols	The columns to display while hovering in the Shiny app. This argument can be: • A dplyr selector (such as dplyr::starts_with()) or a set of selector if they are enclosed with in c(). • A character vector.
hover_only	A logical to determine if interactive highlighting of points is enabled (the default) or not. This can be helpful for very large data sets.

Details

For resampling methods that produce more than one hold-out prediction per row (e.g. the bootstrap, repeated V-fold cross-validation), the predicted values shown in the plots are averages of the predictions for that specific row.

The ggplot2 theme used in the Shiny application corresponds to the current theme in the R session. Run ggplot2::theme_set() to change the theme for the plots in the Shiny application.

For classification models, there is a toggle on the bottom left of the application to choose between "Unscaled (i.e. linear)" and "Logit scaled" probability scaling. The first options plots the raw probabilities while the logit scaling uses scales::logit_trans() to rescale the axis. This can be helpful when a model with a linear predictor is used (e.g. logistic or multinomial regression) since it can show linear effects from a feature more easily.

When using the application, there may be warnings printed in the console about "event tied a source ID ... not registered". These can be ignored.

When racing results are explored, the shiny application will only allow tuning parameter combinations that were fully resampled. As a result, parameter combinations that were discarded during the race will now be able to be selected.

Value

A shiny application.

Examples

```
data(ames_mlp_itr)

if (interactive()) {
  explore(ames_mlp_itr, hover_cols = dplyr::contains("tude"))
}
```

scat_fda_bt *Tuned flexible discriminant analysis results*

Description

This object has the results when a flexible discriminant analysis model was tuned over the interaction degree parameters.

Details

To reduce the object size, five bootstraps were used for resampling and missing data were removed.

The code used to produce this object:

```
library(tidymodels)
library(discrim)
tidymodels_prefer()

# ------------------------------------------------------------------------------

ctrl_gr <- control_grid(save_pred = TRUE)

# ------------------------------------------------------------------------------

data(scat)
scat <- scat[complete.cases(scat), ]

# ------------------------------------------------------------------------------

set.seed(1)
scat_rs <- bootstraps(scat, times = 5)

scat_fda_bt <-
  discrim_flexible(prod_degree = tune()) %>%
  tune_grid(
    Species ~ .,
    resamples = scat_rs,
    control = ctrl_gr
  )
```

Value

An object with primary class `tune_results`.

two_class_final *Test set results for logistic regression*

Description

This object has the results when a logistic regression model is fit to the training set and is evaluated on the test set.

Details

The code used to produce this object:

```
library(tidymodels)
tidymodels_prefer()

# ------------------------------------------------------------------------------

set.seed(1)
data(two_class_dat)

# ------------------------------------------------------------------------------

two_class_split <- initial_split(two_class_dat)
# ------------------------------------------------------------------------------

glm_spec <- logistic_reg()

two_class_final <-
  glm_spec %>%
  last_fit(
    Class  ~ .,
    split = two_class_split
  )
```

Value

An object with primary class `last_fit`.

Index