Model Butcher

Version 0.1.5

Description Provides a set of five S3 generics to axe
components of fitted model objects and help reduce the size of model
objects saved to disk.

License MIT + file LICENSE

URL https://butcher.tidymodels.org

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

Depends R (>= 3.1)

Imports fs,
lobstr (>= 1.1.1),
methods,
purrr,
rlang,
tibble (>= 2.1.1),
usethis (>= 1.5.0),
utils

Suggests C50,
caret,
clisymbols,
covr,
ddalpha,
dimRed,
dplyr,
e1071,
earth,
fastICA,
flexsurv,
ipred,
kernlab,
kknn,
knitr,
MASS,
mda,
modeldata,
NMF,
met,
R topics documented:

parsnip (\texttt{\textcopyright} 0.1.6), pkgload, pls, QSARdata, randomForest, ranger, RANN, recipes, rmarkdown, rpart, rsample, RSpectra, survival (\texttt{\textcopyright} = 3.2-10), testthat (\texttt{\textcopyright} = 2.1.0), TH.data, xgboost (\texttt{\textcopyright} = 1.3.2.1)

\textbf{VignetteBuilder} knitr

\textbf{Encoding} UTF-8

\textbf{RoxygenNote} 7.1.1

\textbf{R topics documented:}

axe-C5.0 ................................................. 3
axe-classbagg ........................................... 4
axe-earth ............................................... 5
axe-elnet ............................................ 6
axe-flexsurvreg ...................................... 7
axe-formula .......................................... 8
axe-function ......................................... 9
axe-gausspr ......................................... 10
axe-glmnet ......................................... 11
axe-kknn ............................................... 12
axe-ksvm ............................................ 13
axe-lm ................................................ 14
axe-mda ............................................... 15
axe-model\_fit ....................................... 16
axe-multnet ........................................ 17
axe-meter ........................................... 18
axe-randomForest .................................... 19
axe-ranger ........................................... 20
axe-recipe ......................................... 22
axe-rpart ........................................... 23
axe-sclass ......................................... 25
axe-spark ........................................... 26
axe-survreg ......................................... 27
axe-survreg.penal ................................... 28
axe-terms ............................................ 29
axe-train ........................................... 30
axe-train.recipe .................................... 31
axe-xgb.Booster ...................................... 32
axe_call ............................................. 34
Axing a C5.0.

Description

C5.0 objects are created from the C50 package, which provides an interface to the C5.0 classification model. The models that can be generated include basic tree-based models as well as rule-based models.

Usage

```r
## S3 method for class 'C5.0'
axe_call(x, verbose = FALSE, ...)

## S3 method for class 'C5.0'
axe_ctrl(x, verbose = FALSE, ...)

## S3 method for class 'C5.0'
axe_fitted(x, verbose = FALSE, ...)
```

Arguments

- `x` A model object.
- `verbose` Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is `FALSE`.
- `...` Any additional arguments related to axing.

Value

Axed C5.0 object.

Examples

```r
# Load libraries
suppressWarnings(suppressMessages(library(parsnip)))
suppressWarnings(suppressMessages(library(rsample)))
suppressWarnings(suppressMessages(library(rpart)))

# Load data
set.seed(1234)
```
split <- initial_split(kyphosis, props = 9/10)
spine_train <- training(split)

# Create model and fit

# Try another model from parsnip

# Create model object from original library

data(mlc_churn)
c5_fit3 <- C5.0(x = mlc_churn[, -20], y = mlc_churn$churn)
out <- butcher(c5_fit3, verbose = TRUE)

axe-classbagg

Axing a classbagg object.

Description

classbagg objects are created from the ipred package, which leverages various resampling and bagging techniques to improve predictive models.

Usage

## S3 method for class 'classbagg'
axe_call(x, verbose = FALSE, ...)

## S3 method for class 'classbagg'
axe_data(x, verbose = FALSE, ...)

## S3 method for class 'classbagg'
axe_env(x, verbose = FALSE, ...)

Arguments

x A model object.

verbose Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is FALSE.

... Any additional arguments related to axing.

Value

Axed classbagg object.
Examples

```r
# Load libraries
suppressWarnings(suppressMessages(library(ipred)))
suppressWarnings(suppressMessages(library(rpart)))
suppressWarnings(suppressMessages(library(MASS)))

# Load data
data("GlaucomaM", package = "TH.data")
classbagg_fit <- bagging(Class ~ ., data = GlaucomaM, coob = TRUE)
out <- butcher(classbagg_fit, verbose = TRUE)

# Fit another model
data("DLBCL", package = "ipred")

mod <- bagging(Gene.Expression ~ MGEc.1 + MGEc.2 + MGEc.3 + MGEc.4 + IPI,
data = DLBCL,
               coob = TRUE)

out <- butcher(mod, verbose = TRUE)
```

axe-earth

Axing an earth object.

Description

earth objects are created from the earth package, which is leveraged to do multivariate adaptive regression splines.

Usage

```r
## S3 method for class 'earth'
axe_call(x, verbose = FALSE, ...)

## S3 method for class 'earth'
axe_data(x, verbose = FALSE, ...)

## S3 method for class 'earth'
axe_fitted(x, verbose = FALSE, ...)
```

Arguments

- `x` A model object.
- `verbose` Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is FALSE.
- `...` Any additional arguments related to axing.

Value

Axed earth object.
Examples

```r
# Load libraries
suppressWarnings(suppressMessages(library(parsnip)))

# Create model and fit
earth_fit <- mars(mode = "regression") %>%
  set_engine("earth") %>%
  fit(Volume ~ ., data = trees)
out <- butcher(earth_fit, verbose = TRUE)

# Another earth model object
suppressWarnings(suppressMessages(library(earth)))
earth_mod <- earth(Volume ~ ., data = trees)
out <- butcher(earth_mod, verbose = TRUE)
```

axe-elnet

Axing an elnet.

Description

elnet objects are created from the `glmnet` package, leveraged to fit generalized linear models via penalized maximum likelihood.

Usage

```r
## S3 method for class 'elnet'
axe_call(x, verbose = FALSE, ...)
```

Arguments

- `x` A model object.
- `verbose` Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is `FALSE`.
- `...` Any additional arguments related to axing.

Value

Axed model object.

Examples

```r
if (rlang::is_installed("glmnet")) {

# Load libraries
suppressWarnings(suppressMessages(library(parsnip)))
suppressWarnings(suppressMessages(library(rsample)))

# Load data
split <- initial_split(mtcars, props = 9/10)
car_train <- training(split)
```
# Create model and fit
```r
elnet_fit <- linear_reg(mixture = 0, penalty = 0.1) %>%
  set_engine("glmnet") %>%
  fit_xy(x = car_train[, 2:11], y = car_train[, 1, drop = FALSE])
```
out <- butcher(elnet_fit, verbose = TRUE)

## Description
flexsurvreg objects are created from the `flexsurv` package. They differ from survreg in that the fitted models are not limited to certain parametric distributions. Users can define their own distribution, or leverage distributions like the generalized gamma, generalized F, and the Royston-Parmar spline model.

### Usage
```r
## S3 method for class 'flexsurvreg'
axe_call(x, verbose = FALSE, ...)
## S3 method for class 'flexsurvreg'
axe_env(x, verbose = FALSE, ...)
```

### Arguments
- `x`: A model object.
- `verbose`: Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is `FALSE`.
- `...`: Any additional arguments related to axing.

### Value
Axed flexsurvreg object.

## Examples
```r
# Load libraries
suppressWarnings(suppressMessages(library(parsnip)));
suppressWarnings(suppressMessages(library(flexsurv)))

# Create model and fit
flexsurvreg_fit <- surv_reg(mode = "regression", dist = "gengamma") %>%
  set_engine("flexsurv") %>%
  fit(Surv(Tstart, Tstop, status) ~ trans, data = bosms3)
out <- butcher(flexsurvreg_fit, verbose = TRUE)
```
Another flexsurvreg model object

```r
wrapped_flexsurvreg <- function() {
  some_junk_in_environment <- runif(1e6)
  fit <- flexsurvreg(Surv(futime, fustat) ~ 1,
                     data = ovarian, dist = "weibull")
  return(fit)
}
```

```r
out <- butcher(wrapped_flexsurvreg(), verbose = TRUE)
```

---

**axe-formula**

## Description

Formulas might capture an environment from the modeling development process that carries objects that will not be used for any post-estimation activities.

## Usage

```r
## S3 method for class 'formula'
axe_env(x, verbose = FALSE, ...)
```

## Arguments

- **x** A model object.
- **verbose** Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is `FALSE`.
- **...** Any additional arguments related to axing.

## Value

Axed formula object.

## Examples

```r
wrapped_formula <- function() {
  some_junk_in_environment <- runif(1e6)
  ex <- as.formula(paste("y ~", paste(LETTERS, collapse = "+")))
  return(ex)
}
```

```r
lobstr::obj_size(wrapped_formula())
lobstr::obj_size(butcher(wrapped_formula()))
```

```r
wrapped_quosure <- function() {
  some_junk_in_environment <- runif(1e6)
  out <- rlang::quo(x)
  return(out)
}
```

```r
lobstr::obj_size(wrapped_quosure())
lobstr::obj_size(butcher(wrapped_quosure()))
```
axe-function

Axing functions.

Description

Functions stored in model objects often have heavy environments and bytecode attached. To avoid breaking any post-estimation functions on the model object, the butchered_function class is not appended.

Usage

```r
## S3 method for class 'function'
axe_env(x, verbose = FALSE, ...)
```

Arguments

- `x` A model object.
- `verbose` Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is `FALSE`.
- `...` Any additional arguments related to axing.

Value

Axed function.

Examples

```r
# Load libraries
suppressWarnings(suppressMessages(library(caret)))

data(iris)
train_data <- iris[, 1:4]
train_classes <- iris[, 5]

train_fit <- train(train_data, train_classes,
                   method = "knn",
                   preProcess = c("center", "scale"),
                   tuneLength = 10,
                   trControl = trainControl(method = "cv"))

out <- axe_env(train_fit$modelInfo$prob, verbose = TRUE)
out <- axe_env(train_fit$modelInfo$levels, verbose = TRUE)
out <- axe_env(train_fit$modelInfo$predict, verbose = TRUE)
```
Description

gausspr objects are created from kernlab package, which provides a means to do classification, regression, clustering, novelty detection, quantile regression and dimensionality reduction. Since fitted model objects from kernlab are S4, the butcher gausspr class is not appended.

Usage

```r
## S3 method for class 'gausspr'
axe_call(x, verbose = FALSE, ...)

## S3 method for class 'gausspr'
axe_data(x, verbose = FALSE, ...)

## S3 method for class 'gausspr'
axe_env(x, verbose = FALSE, ...)

## S3 method for class 'gausspr'
axe_fitted(x, verbose = FALSE, ...)
```

Arguments

- `x` A model object.
- `verbose` Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is `FALSE`.
- `...` Any additional arguments related to axing.

Value

Axed gausspr object.

Examples

```r
suppressWarnings(suppressMessages(library(kernlab)))
test <- gausspr(Species ~ ., data = iris, var = 2)
out <- butcher(test, verbose = TRUE)

# Example with simulated regression data
x <- seq(-20, 20, 0.1)
y <- sin(x)/x + rnorm(401, sd = 0.03)
test2 <- gausspr(x, y)
out <- butcher(test2, verbose = TRUE)
```
Description

glmnet objects are created from the glmnet package, leveraged to fit generalized linear models via penalized maximum likelihood.

Usage

## S3 method for class 'glmnet'
axe_call(x, verbose = FALSE, ...)

Arguments

x A model object.
verbose Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is FALSE.
...
Any additional arguments related to axing.

Value

Axed glmnet object.

Examples

if (rlang::is_installed("glmnet")) {
  suppressWarnings(suppressMessages(library(parsnip))

  # Wrap a parsnip glmnet model
  wrapped_parsnip_glmnet <- function() {
    some_junk_in_environment <- runif(1e6)
    model <- logistic_reg(penalty = 10, mixture = 0.1) %>%
               set_engine("glmnet") %>%
               fit(as.factor(vs) ~ ., data = mtcars)
    return(model$fit)
  }

  out <- butcher(wrapped_parsnip_glmnet(), verbose = TRUE)
}
Description

kknn objects are created from the kknn package, which is utilized to do weighted k-Nearest Neighbors for classification, regression and clustering.

Usage

## S3 method for class 'kknn'
axe_call(x, verbose = FALSE, ...)

## S3 method for class 'kknn'
axe_env(x, verbose = FALSE, ...)

## S3 method for class 'kknn'
axe_fitted(x, verbose = FALSE, ...)

Arguments

x A model object.
verbose Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is FALSE.
... Any additional arguments related to axing.

Value

Axed kknn object.

Examples

# Load libraries
suppressWarnings(suppressMessages(library(parsnip)))
suppressWarnings(suppressMessages(library(rsample)))
suppressWarnings(suppressMessages(library(rpart)))
suppressWarnings(suppressMessages(library(kknn)))

# Load data
set.seed(1234)
split <- initial_split(kyphosis, props = 9/10)
spine_train <- training(split)

# Create model and fit
kknn_fit <- nearest_neighbor(mode = "classification",
                           neighbors = 3,
                           weight_func = "gaussian",
                           dist_power = 2) %>%
  set_engine("kknn") %>%
  fit(Kyphosis ~ ., data = spine_train)

out <- butcher(kknn_fit, verbose = TRUE)


# Another kknn model object
m <- dim(iris)[1]
val <- sample(1:m, size = round(m/3), replace = FALSE, prob = rep(1/m, m))
iris.learn <- iris[-val,]
iris.valid <- iris[val,]
kknn_fit <- kknn(Species ~ ., iris.learn, iris.valid, distance = 1, kernel = "triangular")
out <- butcher(kknn_fit, verbose = TRUE)

---

**axe-ksvm**

*Axing a ksvm object.*

**Description**

ksvm objects are created from `kernlab` package, which provides a means to do classification, regression, clustering, novelty detection, quantile regression and dimensionality reduction. Since fitted model objects from `kernlab` are S4, the `butcher.ksvm` class is not appended.

**Usage**

```r
## S3 method for class 'ksvm'
axe_call(x, verbose = FALSE, ...)
```

```r
## S3 method for class 'ksvm'
axe_data(x, verbose = FALSE, ...)
```

```r
## S3 method for class 'ksvm'
axe_fitted(x, verbose = FALSE, ...)
```

**Arguments**

- **x**
  A model object.

- **verbose**
  Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is `FALSE`.

- **...**
  Any additional arguments related to axing.

**Value**

Axed ksvm object.
Examples

# Load libraries
suppressWarnings(suppressMessages(library(parsnip)))
suppressWarnings(suppressMessages(library(kernlab)))

# Load data
data(spam)

# Create model and fit
ksvm_class <- svm_poly(mode = "classification") %>%
  set_engine("kernlab") %>%
  fit(type ~ ., data = spam)

out <- butcher(ksvm_class, verbose = TRUE)

axe-lm

Axing an lm.

Description

lm objects are created from the base stats package.

Usage

## S3 method for class 'lm'
axe_call(x, verbose = FALSE, ...)

## S3 method for class 'lm'
axe_env(x, verbose = FALSE, ...)

## S3 method for class 'lm'
axe_fitted(x, verbose = FALSE, ...)

Arguments

x A model object.

verbose Print information each time an axe method is executed. Notes how much
memory is released and what functions are disabled. Default is FALSE.

... Any additional arguments related to axing.

Value

Axed lm object.

Examples

# Load libraries
suppressWarnings(suppressMessages(library(parsnip)))
suppressWarnings(suppressMessages(library(rsample)))
# Load data
split <- initial_split(mtcars, props = 9/10)
car_train <- training(split)

# Create model and fit
lm_fit <- linear_reg() %>%
  set_engine("lm") %>%
  fit(mpg ~ ., data = car_train)

out <- butcher(lm_fit, verbose = TRUE)

# Another lm object
wrapped_lm <- function() {
  some_junk_in_environment <- runif(1e6)
  fit <- lm(mpg ~ ., data = mtcars)
  return(fit)
}

# Remove junk
cleaned_lm <- axe_env(wrapped_lm(), verbose = TRUE)

# Check size
lobstr::obj_size(cleaned_lm)

# Compare environment in terms component
lobstr::obj_size(attr(wrapped_lm()$terms, ".Environment"))
lobstr::obj_size(attr(cleaned_lm$terms, ".Environment"))

---

**axe-mda**  
*Axing a mda.*

**Description**

mda objects are created from the `mda` package, leveraged to carry out mixture discriminant analysis.

**Usage**

### S3 method for class 'mda'

`axe_call(x, verbose = FALSE, ...)`

### S3 method for class 'mda'

`axe_env(x, verbose = FALSE, ...)`

### S3 method for class 'mda'

`axe_fitted(x, verbose = FALSE, ...)`

**Arguments**

- `x`  
  A model object.

- `verbose`  
  Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is `FALSE`.

- `...`  
  Any additional arguments related to axing.
Value
Axed mda object.

Examples

```r
suppressWarnings(suppressMessages(library(mda)))

fit <- mda(Species ~ ., data = iris)
out <- butcher(fit, verbose = TRUE)

# Another mda object
data(glass)
wrapped_mda <- function() {
  some_junk_in_environment <- runif(1e6)
  fit <- mda(Type ~ ., data = glass)
  return(fit)
}

lobstr::obj_size(wrapped_mda())
lobstr::obj_size(butcher(wrapped_mda()))
```

Description
model_fit objects are created from the parsnip package.

Usage

```r
## S3 method for class 'model_fit'
axe_call(x, verbose = FALSE, ...)

## S3 method for class 'model_fit'
axe_ctrl(x, verbose = FALSE, ...)

## S3 method for class 'model_fit'
axe_data(x, verbose = FALSE, ...)

## S3 method for class 'model_fit'
axe_env(x, verbose = FALSE, ...)

## S3 method for class 'model_fit'
axe_fitted(x, verbose = FALSE, ...)
```

Arguments

- `x`: A model object.
- `verbose`: Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is `FALSE`.
- `...`: Any additional arguments related to axing.
axe-multnet

Value

Axed model_fit object.

Examples

suppressWarnings(suppressMessages(library(parsnip)))
suppressWarnings(suppressMessages(library(rpart)))

# Create model and fit
lm_fit <- linear_reg() %>%
  set_engine("lm") %>%
  fit(mpg ~ ., data = mtcars)
out <- butcher(lm_fit, verbose = TRUE)

# Another parsnip model
rpart_fit <- decision_tree(mode = "regression") %>%
  set_engine("rpart") %>%
  fit(mpg ~ ., data = mtcars, minsplit = 5, cp = 0.1)
out <- butcher(rpart_fit, verbose = TRUE)

---

axe-multnet

Axing an multnet.

Description

multnet objects are created from carrying out multinomial regression in the glmnet package.

Usage

## S3 method for class 'multnet'
axecall(x, verbose = FALSE, ...)

Arguments

x
A model object.

verbose
Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is FALSE.

...
Any additional arguments related to axing.

Value

Axed multnet object.
Examples

```r
if (rlang::is_installed("glmnet")) {
  # Load libraries
  suppressWarnings(suppressMessages(library(parsnip)))
  # Load data
  set.seed(1234)
  predictrs <- matrix(rnorm(100*20), ncol = 20)
  colnames(predictrs) <- paste0("a", seq_len(ncol(predictrs)))
  response <- as.factor(sample(1:4, 100, replace = TRUE))
  # Create model and fit
  multnet_fit <- multinom_reg(penalty = 0.1) %>%
    set_engine("glmnet") %>%
    fit_xy(x = predictrs, y = response)
  out <- butcher(multnet_fit, verbose = TRUE)
}
```

axe-nnet

Axing a nnet.

Description

nnet objects are created from the nnet package, leveraged to fit multilayer perceptron models.

Usage

```r
## S3 method for class 'nnet'
axe_call(x, verbose = FALSE, ...)

## S3 method for class 'nnet'
axe_env(x, verbose = FALSE, ...)

## S3 method for class 'nnet'
axe_fitted(x, verbose = FALSE, ...)
```

Arguments

- `x` A model object.
- `verbose` Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is FALSE.
- `...` Any additional arguments related to axing.

Value

Axed nnet object.
Examples

```r
# Load libraries
suppressWarnings(suppressMessages(library(parsnip)));
suppressWarnings(suppressMessages(library(nnet)))

# Create and fit model
nnet_fit <- mlp("classification", hidden_units = 2) %>%
  set_engine("nnet") %>%
  fit(Species ~ ., data = iris)

out <- butcher(nnet_fit, verbose = TRUE)

# Another nnet object
targets <- class.ind(c(rep("setosa", 50),
                      rep("versicolor", 50),
                      rep("virginica", 50)))

fit <- nnet(iris[,1:4],
            targets,
            size = 2,
            rang = 0.1,
            decay = 5e-4,
            maxit = 20)

out <- butcher(fit, verbose = TRUE)
```

axe-randomForest  
Axing an randomForest.

Description

randomForest objects are created from the randomForest package, which is used to train random forests based on Breiman’s 2001 work. The package supports ensembles of classification and regression trees.

Usage

```r
## S3 method for class 'randomForest'
axe_call(x, verbose = FALSE, ...)

## S3 method for class 'randomForest'
axe_ctrl(x, verbose = FALSE, ...)

## S3 method for class 'randomForest'
axe_env(x, verbose = FALSE, ...)
```

Arguments

- **x**  
  A model object.

- **verbose**  
  Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is FALSE.

- **...**  
  Any additional arguments related to axing.
Value

Axed randomForest object.

Examples

```r
# Load libraries
suppressWarnings(suppressMessages(library(parsnip)))
suppressWarnings(suppressMessages(library(rsample)))
suppressWarnings(suppressMessages(library(rpart)))
suppressWarnings(suppressMessages(library(randomForest)))

# Load data
set.seed(1234)
split <- initial_split(kyphosis, props = 9/10)
spine_train <- training(split)

# Create model and fit
randomForest_fit <- rand_forest(mode = "classification",
  mtry = 2,
  trees = 2,
  min_n = 3) %>%
  set_engine("randomForest") %>%
  fit_xy(x = spine_train[,2:4], y = spine_train$Kyphosis)
out <- butcher(randomForest_fit, verbose = TRUE)

# Another randomForest object
wrapped_rf <- function() {
  some_junk_in_environment <- runif(1e6)
  randomForest_fit <- randomForest(mpg ~ ., data = mtcars)
  return(randomForest_fit)
}

# Remove junk
cleaned_rf <- axe_env(wrapped_rf(), verbose = TRUE)

# Check size
lobstr::obj_size(cleaned_rf)
```

Description

ranger objects are created from the ranger package, which is used as a means to quickly train random forests. The package supports ensembles of classification, regression, survival and probability prediction trees. Given the reliance of post processing functions on the model object, like importance, pvalues and treeInfo, on the first class listed, the butcher_ranger class is not appended.
Usage

## S3 method for class 'ranger'
axe_call(x, verbose = FALSE, ...)

## S3 method for class 'ranger'
axe_fitted(x, verbose = FALSE, ...)

Arguments

x
A model object.

verbose
Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is FALSE.

...
Any additional arguments related to axing.

Value

Axed ranger object.

Examples

# Load libraries
suppressWarnings(suppressMessages(library(parsnip)))
suppressWarnings(suppressMessages(library(rsample)))
suppressWarnings(suppressMessages(library(ranger)))

# Load data
set.seed(1234)
split <- initial_split(iris, props = 9/10)
iris_train <- training(split)

# Create model and fit
ranger_fit <- rand_forest(mode = "classification",
mtry = 2,
trees = 20,
min_n = 3) %>%
set_engine("ranger") %>%
fit(Species ~ ., data = iris_train)

out <- butcher(ranger_fit, verbose = TRUE)

# Another ranger object
wrapped_ranger <- function() {
  n <- 100
  p <- 400
  dat <- data.frame(y = factor(rbinom(n, 1, .5)), replicate(p, runif(n)))
  fit <- ranger(y ~ ., dat, importance = "impurity_corrected")
  return(fit)
}

cleaned_ranger <- axe_fitted(wrapped_ranger(), verbose = TRUE)
Description

Recipe objects are created from the `recipes` package, which is leveraged for its set of data pre-processing tools. These recipes work by sequentially defining each pre-processing step. The implementation of each step, however, results its own class so we bundle all the axe methods related to recipe objects in general here. Note that the butchered class is only added to the recipe as a whole, and not to each pre-processing step.

Usage

```r
## S3 method for class 'recipe'
axe_env(x, verbose = FALSE, ...)

## S3 method for class 'step'
axe_env(x, ...)

## S3 method for class 'step_arrange'
axe_env(x, ...)

## S3 method for class 'step_filter'
axe_env(x, ...)

## S3 method for class 'step_mutate'
axe_env(x, ...)

## S3 method for class 'step_slice'
axe_env(x, ...)

## S3 method for class 'step_impute_bag'
axe_env(x, ...)

## S3 method for class 'step_bagimpute'
axe_env(x, ...)

## S3 method for class 'step_impute_knn'
axe_env(x, ...)

## S3 method for class 'step_knnimpute'
axe_env(x, ...)

## S3 method for class 'step_geodist'
axe_env(x, ...)

## S3 method for class 'step_interact'
axe_env(x, ...)

## S3 method for class 'step_ratio'
axe_env(x, ...)
```
## S3 method for class 'quo'
axe_env(x, ...)

**Arguments**

- **x**: A model object.
- **verbose**: Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is `FALSE`.
- **...**: Any additional arguments related to axing.

**Value**

Axed recipe object.

**Examples**

```r
suppressWarnings(suppressMessages(library(recipes)))
library(modeldata)

data(biomass)

biomass_tr <- biomass[biomass$dataset == "Training",]
rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur, 
data = biomass_tr) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors()) %>%
  step_spatialsign(all_predictors())

out <- butcher(rec, verbose = TRUE)

# Another recipe object
wrapped_recipes <- function() {
  some_junk_in_environment <- runif(1e6)
  return(
    recipe(mpg ~ cyl, data = mtcars) %>%
      step_center(all_predictors()) %>%
      step_scale(all_predictors())
  )
}

# Remove junk
cleaned_recipes <- axe_env(wrapped_recipes(), verbose = TRUE)

# Check size
lobstr::obj_size(cleaned_recipes)
```

---

### axe-rpart

_Axing a rpart._

**Description**

rpart objects are created from the `rpart` package, which is used for recursive partitioning for classification, regression and survival trees.
Usage

## S3 method for class 'rpart'
axe_call(x, verbose = FALSE, ...)

## S3 method for class 'rpart'
axe_ctrl(x, verbose = FALSE, ...)

## S3 method for class 'rpart'
axe_data(x, verbose = FALSE, ...)

## S3 method for class 'rpart'
axe_env(x, verbose = FALSE, ...)

Arguments

- **x**: A model object.
- **verbose**: Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is FALSE.
- **...**: Any additional arguments related to axing.

Value

Axed rpart object.

Examples

```r
# Load libraries
suppressWarnings(suppressMessages(library(parsnip)))
suppressWarnings(suppressMessages(library(rsample)))
suppressWarnings(suppressMessages(library(rpart)))
suppressWarnings(library(lobstr))

# Load data
set.seed(1234)
split <- initial_split(mtcars, props = 9/10)
car_train <- training(split)

# Create model and fit
rpart_fit <- decision_tree(mode = "regression") %>%
  set_engine("rpart") %>%
  fit(mpg ~ ., data = car_train, minsplit = 5, cp = 0.1)
out <- butcher(rpart_fit, verbose = TRUE)

# Another rpart object
wrapped_rpart <- function() {
  some_junk_in_environment <- runif(1e6)
  fit <- rpart(Kyphosis ~ Age + Number + Start,
               data = kyphosis,
               x = TRUE, y = TRUE)
  return(fit)
}

# Remove junk
cleaned_rpart <- axe_env(wrapped_rpart(), verbose = TRUE)
```
axe-sclass

Axing a sclass object.

Description

sclass objects are byproducts of classbagg objects.

Usage

## S3 method for class 'sclass'
axe_call(x, verbose = FALSE, ...)

## S3 method for class 'sclass'
axe_env(x, verbose = FALSE, ...)

Arguments

x  
A model object.

verbose  
Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is FALSE.

...  
Any additional arguments related to axing.

Value

Axed sclass object.

Examples

# Load libraries
suppressWarnings(suppressMessages(library(ipred)))
suppressWarnings(suppressMessages(library(rpart)))
suppressWarnings(suppressMessages(library(MASS)))

# Load data
data("GlaucomaM", package = "TH.data")

classbagg_fit <- bagging(Class ~ ., data = GlaucomaM, coob = TRUE)
out <- butcher(classbagg_fit$mtrees[[1]], verbose = TRUE)

# Another classbagg object
wrapped_classbagg <- function() {
  some_junk_in_environment <- runif(1e6)
  fit <- bagging(Species ~ .,
                 data = iris,
                 nbagg = 10,
                 coob = TRUE)

  return(fit)
# Remove junk

```r
cleaned_classbagg <- butcher(wrapped_classbagg(), verbose = TRUE)
```

# Check size

```r
lobstr::obj_size(cleaned_classbagg)
```

## Description

spark objects are created from the `sparklyr` package, a R interface for Apache Spark. The axe methods available for spark objects are designed such that interoperability is maintained. In other words, for a multilingual machine learning team, butchered spark objects instantiated from `sparklyr` can still be serialized to disk, work in Python, be deployed on Scala, etc. It is also worth noting here that spark objects created from `sparklyr` have a lot of metadata attached to it, including but not limited to the formula, dataset, model, index labels, etc. The axe functions provided are for parsing down the model object both prior saving to disk, or loading from disk. Traditional R save functions are not available for these objects, so functionality is provided in `sparklyr::ml_save`. This function gives the user the option to keep either the pipeline model or the pipeline, so both of these objects are retained from butchering, yet removal of one or the other might be conducive to freeing up memory on disk.

## Usage

```r
## S3 method for class 'ml_model'
axe_call(x, verbose = FALSE, ...)

## S3 method for class 'ml_model'
axe_ctrl(x, verbose = FALSE, ...)

## S3 method for class 'ml_model'
axe_data(x, verbose = FALSE, ...)

## S3 method for class 'ml_model'
axe_fitted(x, verbose = FALSE, ...)
```

## Arguments

- **x**  
  A model object.

- **verbose**  
  Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is `FALSE`.

- **...**  
  Any additional arguments related to axing.

## Value

Axed spark object.
Examples

```r
## Not run:
if (FALSE) {

suppressWarnings(suppressMessages(library(sparklyr)))

sc <- spark_connect(master = "local")

iris_tbls <- sdf_copy_to(sc, iris, overwrite = TRUE) %>%
  sdf_random_split(train = 2/3, validation = 2/3, seed = 2018)

train <- iris_tbls$train
spark_fit <- ml_logistic_regression(train, Species ~ .)

out <- butcher(spark_fit, verbose = TRUE)

spark_disconnect(sc)
}
## End(Not run)
```

Description

`survreg` objects are created from the `survival` package. They are returned from the `survreg` function, representing fitted parametric survival models.

Usage

```r
## S3 method for class 'survreg'
axe_call(x, verbose = FALSE, ...)

## S3 method for class 'survreg'
axe_data(x, verbose = FALSE, ...)

## S3 method for class 'survreg'
axe_env(x, verbose = FALSE, ...)
```

Arguments

- **x**
  - A model object.

- **verbose**
  - Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is `FALSE`.

- **...**
  - Any additional arguments related to axing.

Value

Axed `survreg` object.
Examples

```r
# Load libraries
suppressWarnings(suppressMessages(library(parsnip)))
suppressWarnings(suppressMessages(library(survival)))

# Create model and fit
survreg_fit <- surv_reg(mode = "regression", dist = "weibull") %>%
  set_engine("survival") %>%
  fit(Surv(futime, fustat) ~ 1, data = ovarian)

out <- butcher(survreg_fit, verbose = TRUE)
```

```r
# Another survreg object
wrapped_survreg <- function() {
  some_junk_in_environment <- runif(1e6)
  fit <- survreg(Surv(time, status) ~ ph.ecog + age + strata(sex),
                  data = lung)
  return(fit)
}

# Remove junk
cleaned_survreg <- butcher(wrapped_survreg(), verbose = TRUE)
```

```r
# Check size
lobstr::obj_size(cleaned_survreg)
```

---

axe-survreg.penal

Axing an survreg.penal

Description

survreg.penal objects are created from the `survival` package. They are returned from the `survreg` function, representing fitted parametric survival models.

Usage

```r
## S3 method for class 'survreg.penal'
axe_call(x, verbose = FALSE, ...)
```

```r
## S3 method for class 'survreg.penal'
axe_data(x, verbose = FALSE, ...)
```

```r
## S3 method for class 'survreg.penal'
axe_env(x, verbose = FALSE, ...)
```

Arguments

- **x**  
  A model object.
- **verbose**  
  Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is `FALSE`.
- **...**  
  Any additional arguments related to axing.
Value

Axed survreg object.

Examples

```r
# Load libraries
suppressWarnings(suppressMessages(library(parsnip)))
suppressWarnings(suppressMessages(library(survival)))
suppressWarnings(library(lobstr))

# Create model and fit
survreg_fit <- surv_reg(mode = "regression", dist = "weibull") %>%
  set_engine("survival") %>%
  fit(Surv(time, status) ~ rx, data = rats)
out <- butcher(survreg_fit, verbose = TRUE)

# Another survreg.penal object
wrapped_survreg.penal <- function() {
  some_junk_in_environment <- runif(1e6)
  fit <- survreg(Surv(time, status) ~ rx,
                data = rats, subset = (sex == "f"))
  return(fit)
}

# Remove junk
cleaned_sp <- axe_env(wrapped_survreg.penal(), verbose = TRUE)

# Check size
lobstr::obj_size(cleaned_sp)
```

axe-terms

Axing for terms inputs.

Description

Generics related to axing objects of the term class.

Usage

```r
## S3 method for class 'terms'
axe_env(x, verbose = FALSE, ...)
```

Arguments

- `x` A model object.
- `verbose` Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is FALSE.
- `...` Any additional arguments related to axing.

Value

Axed terms object.
Examples

```r
# Using lm
wrapped_lm <- function() {
    some_junk_in_environment <- runif(1e6)
    fit <- lm(mpg ~ ., data = mtcars)
    return(fit)
}

# Remove junk
cleaned_lm <- axe_env(wrapped_lm(), verbose = TRUE)

# Check size
lobstr::obj_size(cleaned_lm)

# Compare environment in terms component
lobstr::obj_size(attr(wrapped_lm()$terms, ".Environment"))
lobstr::obj_size(attr(cleaned_lm$terms, ".Environment"))

# Using rpart
suppressWarnings(library(rpart))

wrapped_rpart <- function() {
    some_junk_in_environment <- runif(1e6)
    fit <- rpart(Kyphosis ~ Age + Number + Start,
                 data = kyphosis,
                 x = TRUE,
                 y = TRUE)
    return(fit)
}

lobstr::obj_size(wrapped_rpart())
lobstr::obj_size(axe_env(wrapped_rpart()))
```

axe-train

Axing a train object.

Description

Train objects are created from the caret package.

Usage

```r
## S3 method for class 'train'
axe_call(x, verbose = FALSE, ...)

## S3 method for class 'train'
axe_ctrl(x, verbose = FALSE, ...)

## S3 method for class 'train'
axe_data(x, verbose = FALSE, ...)
```
axe-train.recipe

## S3 method for class 'train'
axe_env(x, verbose = FALSE, ...)

## S3 method for class 'train'
axe_fitted(x, verbose = FALSE, ...)

### Arguments

- **x**: A model object.
- **verbose**: Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is FALSE.
- **...**: Any additional arguments related to axing.

### Value

Axed train object.

### Examples

```r
# Load libraries
suppressWarnings(suppressMessages(library(caret)))

data(iris)
train_data <- iris[, 1:4]
train_classes <- iris[, 5]

train_fit <- train(train_data, train_classes,
  method = "knn",
  preProcess = c("center", "scale"),
  tuneLength = 10,
  trControl = trainControl(method = "cv"))

out <- butcher(train_fit, verbose = TRUE)
```

---

### Description

train.recipe objects are slightly different from train objects created from the caret package in that it also includes instructions from a recipe for data pre-processing. Axing functions specific to train.recipe are thus included as additional steps are required to remove parts of train.recipe objects.

### Usage

```r
## S3 method for class 'train.recipe'
axe_call(x, ...)

## S3 method for class 'train.recipe'
axe_ctrl(x, ...)
```
### axe-xgb.Booster

Axing a xgb.Booster.

**Description**

xgb.Booster objects are created from the `xgboost` package, which provides efficient and scalable implementations of gradient boosted decision trees. Given the reliance of post-processing functions on the model object, like `xgb.Booster.complete`, on the first class listed, the `butcher_xgb.Booster` class is not appended.
## axe-xgb.Booster

### Usage

```r
## S3 method for class 'xgb.Booster'
axe_call(x, verbose = FALSE, ...)

## S3 method for class 'xgb.Booster'
axe_ctrl(x, verbose = FALSE, ...)

## S3 method for class 'xgb.Booster'
axe_env(x, verbose = FALSE, ...)

## S3 method for class 'xgb.Booster'
axe_fitted(x, verbose = FALSE, ...)
```

### Arguments

- `x` A model object.
- `verbose` Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is `FALSE`.
- `...` Any additional arguments related to axing.

### Value

Axed xgb.Booster object.

### Examples

```r
suppressWarnings(suppressMessages(library(xgboost)))
suppressWarnings(suppressMessages(library(parsnip)))
data(agaricus.train)
bst <- xgboost(data = agaricus.train$data,
                 label = agaricus.train$label,
                 eta = 1,
                 nthread = 2,
                 nrounds = 2,
                 eval_metric = "logloss",
                 objective = "binary:logistic",
                 verbose = 0)

out <- butcher(bst, verbose = TRUE)

# Another xgboost model
fit <- boost_tree(mode = "classification", trees = 20) %>%
      set_engine("xgboost", eval_metric = "mlogloss") %>%
      fit(Species ~ ., data = iris)

out <- butcher(fit, verbose = TRUE)
```
**Description**

Replace the call object attached to modeling objects with a placeholder.

**Usage**

```r
text = axe_call(x, verbose = FALSE, ...)
```

**Arguments**

- `x`: A model object.
- `verbose`: Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is `FALSE`.
- `...`: Any additional arguments related to axing.

**Value**

Model object without call attribute.

**Methods**

See the following help topics for more details about individual methods:

- `butcher`
  - `axe-C5.0`: C5.0
  - `axe-classbagg`: classbagg
  - `axe-earth`: earth
  - `axe-elnet`: elnet
  - `axe-flexsurvreg`: flexsurvreg
  - `axe-gausspr`: gausspr
  - `axe-glmnet`: glmnet
  - `axe-kknn`: kknn
  - `axe-ksvm`: ksvm
  - `axe-lm`: lm
  - `axe-mda`: mda
  - `axe-model_fit`: model_fit
  - `axe-multnet`: multnet
  - `axe-nnet`: nnet
  - `axe-randomForest`: randomForest
  - `axe-ranger`: ranger
  - `axe-rpart`: rpart
  - `axe-sclass`: sclass
  - `axe-spark`: ml_model
*axe_ctrl*

- *axe-survreg*: survreg
- *axe-survreg.penal*: survreg.penal
- *axe-train*: train
- *axe-train.recipe*: train.recipe

---

**Description**

Remove the controls from training attached to modeling objects.

**Usage**

```r
axe_ctrl(x, verbose = FALSE, ...)
```

**Arguments**

- **x**: A model object.
- **verbose**: Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is FALSE.
- **...**: Any additional arguments related to axing.

**Value**

Model object without control tuning parameters from training.

**Methods**

See the following help topics for more details about individual methods:

- *butcher*
  - *axe-C5.0*: C5.0
  - *axe-model.fit*: model_fit
  - *axe-randomForest*: randomForest
  - *axe-rpart*: rpart
  - *axe-spark*: ml_model
  - *axe-train*: train
  - *axe-train.recipe*: train.recipe
Description

Remove the training data attached to modeling objects.

Usage

\[
\text{axe\_data}(x, \text{verbose} = \text{FALSE}, \ldots)
\]

Arguments

- **x**: A model object.
- **verbose**: Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is `FALSE`.
- **...**: Any additional arguments related to axing.

Value

Model object without the training data

Methods

See the following help topics for more details about individual methods:

- **butcher**
  - `axe-classbagg`: classbagg
  - `axe-earth`: earth
  - `axe-gausspr`: gausspr
  - `axe-ksvm`: ksvm
  - `axe-model_fit`: model_fit
  - `axe-rpart`: rpart
  - `axe-spark`: ml_model
  - `axe-survreg`: survreg
  - `axe-survreg.penal`: survreg.penal
  - `axe-train`: train
  - `axe-train.recipe`: train.recipe
Axe an environment.

Description

Remove the environment(s) attached to modeling objects as they are not required in the
downstream analysis pipeline. If found, the environment is replaced with `rlang::base_env()`.

Usage

`axe_env(x, verbose = FALSE, ...)`

Arguments

- `x` A model object.
- `verbose` Print information each time an axe method is executed. Notes how much
  memory is released and what functions are disabled. Default is `FALSE`.
- `...` Any additional arguments related to axing.

Value

Model object with empty environments.

Methods

See the following help topics for more details about individual methods:

- `butcher`
  - `axe-classbagg`: classbagg
  - `axe-flexsurvreg`: flexsurvreg
  - `axe-formula`: formula
  - `axe-function`: function
  - `axe-gausspr`: gausspr
  - `axe-kknn`: kknn
  - `axe-lm`: lm
  - `axe-mda`: mda
  - `axe-model_fit`: model_fit
  - `axe-nnet`: nnet
  - `axe-randomForest`: randomForest
  - `axe-recipe`: quosure, recipe, step, step_arrange, step_bagimpute, step_filter,
    step_geodist, step_impute_bag, step_impute_knn, step_interact, step_knnimpute, step_mutate,
    step_ratio, step_slice
  - `axe-rpart`: rpart
  - `axe-sclass`: sclass
  - `axe-survreg`: survreg
  - `axe-survreg.penal`: survreg.penal
axe_fitted

Description
Remove the fitted values attached to modeling objects.

Usage
axe_fitted(x, verbose = FALSE, ...)

Arguments
x A model object.
verbose Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is FALSE.
... Any additional arguments related to axing.

Value
Model object without the fitted values.

Methods
See the following help topics for more details about individual methods:
butcher
axe-C5.0: C5.0
axe-earth: earth
axe-gausspr: gausspr
axe-kknn: kknn
axe-ksvm: ksvm
axe-lm: lm
axe-mda: mda
axe-model_fit: model_fit
axe-nnet: nnet
axe-ranger: ranger
axe-spark: ml_model
axe-train: train
axe-train.recipe: train.recipe
butcher

*Butcher an object.*

**Description**

Reduce the size of a model object so that it takes up less memory on disk. Currently, the model object is stripped down to the point that only the minimal components necessary for the `predict` function to work remain. Future adjustments to this function will be needed to avoid removal of model fit components to ensure it works with other downstream functions.

**Usage**

```r
butcher(x, verbose = FALSE, ...)
```

**Arguments**

- `x` A model object.
- `verbose` Print information each time an axe method is executed. Notes how much memory is released and what functions are disabled. Default is `FALSE`.
- `...` Any additional arguments related to axing.

**Value**

Axed model object with new butcher subclass assignment.

---

**butcher_example**

*Get path to model object example.*

**Description**

butcher comes bundled with some example files in its ‘inst/extdata’ directory. This function was copied from readxl and placed here to make the instantiated model objects easy to access.

**Usage**

```r
butcher_example(path = NULL)
```

**Arguments**

- `path` Name of file. If ‘NULL’, the example files will be listed.
locate

Locate part of an object.

Description

Locate where a specific component of an object might exist within the model object itself. This function is restricted in that only items that can be axed can be found.

Usage

locate(x, name = NULL)

Arguments

x
A model object.

name
A name associated with model component of interest. This defaults to NULL. Possible components include: env, call, data, ctrl, and fitted.

Value

Location of specific component in a model object.

Examples

lm_fit <- lm(mpg ~ ., data = mtcars)
locate(lm_fit, name = "env")
locate(lm_fit, name = "call")

new_model_butcher

New axe functions for a modeling object.

Description

new_model_butcher() will instantiate the following to help us develop new axe functions around removing parts of a new modeling object:

- Add modeling package to Suggests
- Generate and populate an axe file under R/
- Generate and populate an test file under testthat/

Usage

new_model_butcher(model_class, package_name, open = interactive())

Arguments

model_class
A string that captures the class name of the new model object.

package_name
A string that captures the package name from which the new model is made.

open
Check if user is in interactive mode, and if so, opens the new files for editing.
ui

Console Messages

Description
These functions leverage the ui.R as provided in the usethis package. Original reference here: https://github.com/r-lib/usethis/blob/master/R/ui.R. These console messages are created such that the user is aware of the effects of removing specific components from the model object.

Usage
memory_released(og, butchered)
assess_object(og, butchered)

Arguments
og Original model object.
butchered Butchered model object.

weigh Weigh the object.

Description
Evaluate the size of each element contained in a model object.

Usage
weigh(x, threshold = 0, units = "MB", ...)

Arguments
x A model object.
threshold The minimum threshold desired for model component size to display.
units The units in which to display the size of each component within the model object of interest. Defaults to MB. Other options include KB and GB.
... Any additional arguments for weighing.

Value
Tibble with weights of object components in decreasing magnitude.

Examples
simulate_x <- matrix(runif(1e+6), ncol = 2)
simulate_y <- runif(dim(simulate_x)[1])
lm_out <- lm(simulate_y ~ simulate_x)
weigh(lm_out)
Index

assess_object (ui), 41
axe-C5.0, 3
axe-classbagg, 4
axe-earth, 5
axe-elnnet, 6
axe-flexsurvreg, 7
axe-formula, 8
axe-function, 9
axe-gausspr, 10
axe-glmnet, 11
axe-kknn, 12
axe-ksvm, 13
axe-lm, 14
axe-mda, 15
axe-model_fit, 16
axe-multnet, 17
axe-nnet, 18
axe-randomForest, 19
axe-ranger, 20
axe-rpart, 23
axe-sclass, 25
axe-spark, 26
axe-survreg, 27
axe-survreg.penal, 28
axe-terms, 29
axe-train, 30
axe-train.recipe, 31
axe-xgb.Booster, 32
axe_call, 34
axe_call.C5.0 (axe-C5.0), 3
axe_call.classbagg (axe-classbagg), 4
axe_call.earth (axe-earth), 5
axe_call.elnet (axe-elnnet), 6
axe_call.flexsurvreg (axe-flexsurvreg), 7
axe_call.gausspr (axe-gausspr), 10
axe_call.glmnet (axe-glmnet), 11
axe_call.kknn (axe-kknn), 12
axe_call.ksvm (axe-ksvm), 13
axe_call.lm (axe-lm), 14
axe_call.mda (axe-mda), 15
axe_call.ml_model (axe-spark), 26
axe_call.model_fit (axe-model_fit), 16
axe_call.multnet (axe-multnet), 17
axe_call.nnet (axe-nnet), 18
axe_call.randomForest (axe-randomForest), 19
axe_call.ranger (axe-ranger), 20
axe_call.rpart (axe-rpart), 23
axe_call.sclass (axe-sclass), 25
axe_call.survreg (axe-survreg), 27
axe_call.survreg.penal (axe-survreg.penal), 28
axe_call.train (axe-train), 30
axe_call.train.recipe (axe-train.recipe), 31
axe_call.xgb.Booster (axe-xgb.Booster), 32
axe_ctrl, 35
axe_ctrl.C5.0 (axe-C5.0), 3
axe_ctrl.ml_model (axe-spark), 26
axe_ctrl.model_fit (axe-model_fit), 16
axe_ctrl.randomForest (axe-randomForest), 19
axe_ctrl.rpart (axe-rpart), 23
axe_ctrl.train (axe-train), 30
axe_ctrl.train.recipe (axe-train.recipe), 31
axe_ctrl.xgb.Booster (axe-xgb.Booster), 32
axe_data, 36
axe_data.classbagg (axe-classbagg), 4
axe_data.earth (axe-earth), 5
axe_data.gausspr (axe-gausspr), 10
axe_data.ksvm (axe-ksvm), 13
axe_data.ml_model (axe-spark), 26
axe_data.model_fit (axe-model_fit), 16
axe_data.rpart (axe-rpart), 23
axe_data.survreg (axe-survreg), 27
axe_data.survreg.penal (axe-survreg.penal), 28
axe_data.train (axe-train), 30
axe_data.train.recipe (axe-train.recipe), 31
axe_env, 37