Package 'mlr3viz'

March 15, 2021

Title Visualizations for 'mlr3'

Version 0.5.3

Description Provides visualizations for 'mlr3' objects such as tasks, predictions, resample results or benchmark results via the autoplot() generic of 'ggplot2'. The returned 'ggplot' objects are intended to provide sensible defaults, yet can easily be customized to create camera-ready figures. Visualizations include barplots, boxplots, histograms, ROC curves, and Precision-Recall curves.

License LGPL-3

URL https://mlr3viz.mlr-org.com, https://github.com/mlr-org/mlr3viz

BugReports https://github.com/mlr-org/mlr3viz/issues

Depends R (>= 3.1.0)

Imports checkmate, data.table, ggplot2 (>= 3.3.0), mlr3misc (>= 0.7.0), utils

Suggests cluster, distr6 (>= 1.4.4), factoextra, GGally, ggfortify (>= 0.4.11), ggparty, glmnet, lgr, mlr3 (>= 0.6.0), mlr3cluster, mlr3filters, mlr3learners, mlr3proba (>= 0.3.2), mlr3tuning (>= 0.7.0), paradox, partykit, patchwork (>= 1.1.1), precrec, ranger, rpart, stats, survival, testthat (>= 3.0.0), xgboost

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

LazyData true

NeedsCompilation no

RoxygenNote 7.1.1

Author Michel Lang [cre, aut] (<https://orcid.org/0000-0001-9754-0393>), Patrick Schratz [aut] (<https://orcid.org/0000-0003-0748-6624>), Raphael Sonabend [aut] (<https://orcid.org/0000-0001-9225-4654>), Marc Becker [aut] (<https://orcid.org/0000-0002-8115-0400>), Damir Pulatov [ctb]

Maintainer Michel Lang <michellang@gmail.com>

Repository CRAN

Date/Publication 2021-03-15 16:50:07 UTC

R topics documented:

`mlr3viz-package` *mlr3viz: Visualizations for 'mlr3'*

Description

Provides visualizations for 'mlr3' objects such as tasks, predictions, resample results or benchmark results via the autoplot() generic of 'ggplot2'. The returned 'ggplot' objects are intended to provide sensible defaults, yet can easily be customized to create camera-ready figures. Visualizations include barplots, boxplots, histograms, ROC curves, and Precision-Recall curves.

Author(s)

Maintainer: Michel Lang <michellang@gmail.com> (ORCID)

Authors:

- Patrick Schratz <patrick.schratz@gmail.com> (ORCID)
- Raphael Sonabend <raphael.sonabend.15@ucl.ac.uk> (ORCID)

- Marc Becker <marcbecker@posteo.de> (ORCID)

Other contributors:

- Damir Pulatov <dpulatov@uwyo.edu> [contributor]

See Also

Useful links:

- https://mlr3viz.mlr-org.com
- https://github.com/mlr-org/mlr3viz
- Report bugs at https://github.com/mlr-org/mlr3viz/issues

`as_precrec` *Convert to 'precrec' Format*

Description

Converts to a format which is understood by `precrec::evalmod()` of package **precrec**.

Usage

```
as_precrec(object)

## S3 method for class 'PredictionClassif'
as_precrec(object)

## S3 method for class 'ResampleResult'
as_precrec(object)

## S3 method for class 'BenchmarkResult'
as_precrec(object)
```

Arguments

`object`	(any) Object to convert.

Value

Object as created by `precrec::mmdata()`.

References

Saito T, Rehmsmeier M (2017). "Precrec: fast and accurate precision-recall and ROC curve calculations in R." *Bioinformatics*, **33**(1), 145-147. doi: 10.1093/bioinformatics/btw570.

`autoplot.BenchmarkResult` *Plot for BenchmarkResult*

Description

Generates plots for mlr3::BenchmarkResult, depending on argument `type`:

- `"boxplot"` (default): Boxplots of performance measures, one box per mlr3::Learner and one facet per mlr3::Task.
- `"roc"`: ROC curve (1 - specificity on x, sensitivity on y). The mlr3::BenchmarkResult may only have a single mlr3::Task and a single mlr3::Resampling. Note that you can subset any mlr3::BenchmarkResult with its $filter() method (see examples). Requires package **precrec**. Additional arguments will be passed down to the respective `autoplot()` function in package **precrec**. Arguments `calc_avg` and `cb_alpha` are passed to `precrec::evalmod()`.
- `"prc"`: Precision recall curve. See `"roc"`.

Usage

```
## S3 method for class 'BenchmarkResult'
autoplot(object, type = "boxplot", measure = NULL, ...)
```

Arguments

`object`	(mlr3::BenchmarkResult).
`type`	(character(1)): Type of the plot. See description.
`measure`	(mlr3::Measure).
`...`	(any): Additional arguments, passed down to the respective `geom` or plotting function.

Value

`ggplot2::ggplot()` object.

References

Saito T, Rehmsmeier M (2017). "Precrec: fast and accurate precision-recall and ROC curve calculations in R." *Bioinformatics*, **33**(1), 145-147. doi: 10.1093/bioinformatics/btw570.

Examples

```
library(mlr3)
library(mlr3viz)

tasks = tsks(c("pima", "sonar"))
learner = lrns(c("classif.featureless", "classif.rpart"),
```

```
  predict_type = "prob")
resampling = rsmps("cv")
object = benchmark(benchmark_grid(tasks, learner, resampling))

head(fortify(object))
autoplot(object)
autoplot(object$clone(deep = TRUE)$filter(task_ids = "pima"), type = "roc")
```

`autoplot.Filter` *Plot for Filter Scores*

Description

Generates plots for mlr3filters::Filter, depending on argument `type`:

- `"barplot"` (default): Bar plot of filter scores.

Usage

```
## S3 method for class 'Filter'
autoplot(object, type = "boxplot", n = Inf, ...)
```

Arguments

`object`	(mlr3filters::Filter).
`type`	(character(1)): Type of the plot. See description.
`n`	(`integer(1)`) Only include the first `n` features with highest importance. Defaults to all features.
`...`	(any): Additional argument, passed down to the respective `geom`.

Value

`ggplot2::ggplot()` object.

Examples

```
library(mlr3)
library(mlr3viz)
library(mlr3filters)

task = tsk("mtcars")
f = flt("correlation")
f$calculate(task)

head(fortify(f))
autoplot(f, n = 5)
```

`autoplot.LearnerClassifCVGlmnet` *Plot for LearnerClassifGlmnet / LearnerRegrGlmnet / LearnerClassifCVGlmnet / LearnerRegrCVGlmnet*

Description

Visualizations for mlr3learners::mlr_learners_classif.glmnet, mlr3learners::mlr_learners_regr.glmnet, mlr3learners::mlr_learners_classif.cv_glmnet and mlr3learners::mlr_learners_regr.cv_glmnet using the package **ggfortify**.

Note that learner-specific plots are experimental and subject to change.

Usage

```
## S3 method for class 'LearnerClassifCVGlmnet'
autoplot(object, ...)

## S3 method for class 'LearnerClassifGlmnet'
autoplot(object, ...)

## S3 method for class 'LearnerRegrCVGlmnet'
autoplot(object, ...)

## S3 method for class 'LearnerRegrGlmnet'
autoplot(object, ...)
```

Arguments

`object`	(mlr3learners::LearnerClassifGlmnet | mlr3learners::LearnerRegrGlmnet | mlr3learners::LearnerRegrCV | mlr3learners::LearnerRegrCVGlmnet).
`...`	(any): Additional arguments, passed down to `ggparty::autoplot.party()`.

Value

`ggplot2::ggplot()` object.

References

Tang Y, Horikoshi M, Li W (2016). "ggfortify: Unified Interface to Visualize Statistical Result of Popular R Packages." *The R Journal*, **8**(2), 474–485. doi: 10.32614/RJ2016060.

Examples

```
## Not run:
library(mlr3)
library(mlr3viz)
library(mlr3learners)
```

```
# classification
task = tsk("sonar")
learner = lrn("classif.glmnet")
learner$train(task)
autoplot(learner)

# regression
task = tsk("mtcars")
learner = lrn("regr.glmnet")
learner$train(task)
autoplot(learner)

## End(Not run)
```

`autoplot.LearnerClassifRpart` *Plot for LearnerClassifRpart / LearnerRegrRpart*

Description

Visualize trees for mlr3::mlr_learners_classif.rpart and mlr3::mlr_learners_regr.rpart using the package **ggparty**.

Contrary to **ggparty**, boxplots are shown in the terminal nodes for regression trees.

Note that learner-specific plots are experimental and subject to change.

Usage

```
## S3 method for class 'LearnerClassifRpart'
autoplot(object, ...)

## S3 method for class 'LearnerRegrRpart'
autoplot(object, ...)
```

Arguments

`object`	(mlr3::LearnerClassifRpart | mlr3::LearnerRegrRpart).
`...`	(any): Additional arguments, passed down to `ggparty::autoplot.party()`.

Value

`ggplot2::ggplot()` object.

Examples

```
library(mlr3)
library(mlr3viz)

# classification
task = tsk("iris")
learner = lrn("classif.rpart", keep_model = TRUE)
learner$train(task)
autoplot(learner)

# regression
task = tsk("mtcars")
learner = lrn("regr.rpart", keep_model = TRUE)
learner$train(task)
autoplot(learner)
```

`autoplot.LearnerClustHierarchical`
Plot for Hierarchical Clustering Learners

Description

Visualize dendrograms for hierarchical clusterers using the package **factoextra**.

Note that learner-specific plots are experimental and subject to change.

Usage

```
## S3 method for class 'LearnerClustHierarchical'
autoplot(object, ...)
```

Arguments

`object`	(mlr3cluster::LearnerClustAgnes | mlr3cluster::LearnerClustDiana).
`...`	(any): Additional arguments, passed down to function `factoextra::fviz_dend()` in package **factoextra**.

Value

`ggplot2::ggplot()` object.

Examples

```
library(mlr3)
library(mlr3cluster)
library(mlr3viz)

task = mlr_tasks$get("usarrests")
```

```
# agnes clustering
learner = mlr_learners$get("clust.agnes")
learner$train(task)
autoplot(learner)

# diana clustering
learner = mlr_learners$get("clust.diana")
learner$train(task)
autoplot(learner,
  k = learner$param_set$values$k, rect_fill = TRUE,
  rect = TRUE, rect_border = "red")
```

`autoplot.PredictionClassif` *Plot for PredictionClassif*

Description

Generates plots for mlr3::PredictionClassif, depending on argument `type`:

- `"stacked"` (default): Stacked barplot of true and estimated class labels.
- `"roc"`: ROC curve (1 - specificity on x, sensitivity on y). Requires package **precrec**.
- `"prc"`: Precision recall curve. Requires package **precrec**.

Usage

```
## S3 method for class 'PredictionClassif'
autoplot(object, type = "stacked", ...)
```

Arguments

`object`	(mlr3::PredictionClassif).
`type`	(character(1)): Type of the plot. See description.
`...`	(any): Additional arguments, passed down to the respective `geom` or plotting function.

Value

`ggplot2::ggplot()` object.

References

Saito T, Rehmsmeier M (2017). "Precrec: fast and accurate precision-recall and ROC curve calculations in R." *Bioinformatics*, **33**(1), 145-147. doi: 10.1093/bioinformatics/btw570.

Examples

```
library(mlr3)
library(mlr3viz)

task = tsk("spam")
learner = lrn("classif.rpart", predict_type = "prob")
object = learner$train(task)$predict(task)

head(fortify(object))
autoplot(object)
autoplot(object, type = "roc")
autoplot(object, type = "prc")
```

---

autoplot.PredictionClust

Plot for PredictionClust

Description

Generates plots for mlr3cluster::PredictionClust, depending on argument `type`:

- `"scatter"` (default): scatterplot with correlation values and colored cluster assignments.
- `"sil"`: Silhouette plot with mean silhouette value as a reference line. Requires package **ggfortify**.
- `"pca"`: Perform PCA on data and color code cluster assignments. Inspired by and uses ggfortify::autoplot.kmeans.

Usage

```
## S3 method for class 'PredictionClust'
autoplot(object, task, row_ids = NULL, type = "scatter", ...)
```

Arguments

`object`	(mlr3cluster::PredictionClust).
`task`	(mlr3cluster::TaskClust).
`row_ids`	row ids to subset task data to ensure that only the data used to make predictions are shown in plots.
`type`	(character(1)): Type of the plot. See description.
`...`	(any): Additional arguments, passed down to the respective `geom`.

Value

`ggplot2::ggplot()` object.

References

Tang Y, Horikoshi M, Li W (2016). "ggfortify: Unified Interface to Visualize Statistical Result of Popular R Packages." *The R Journal*, **8**(2), 474–485. doi: 10.32614/RJ2016060.

Examples

```
library(mlr3)
library(mlr3cluster)
library(mlr3viz)

task = tsk("usarrests")
learner = lrn("clust.kmeans", centers = 3)
object = learner$train(task)$predict(task)

head(fortify(object))
autoplot(object, task)
```

---

autoplot.PredictionRegr

Plot for PredictionRegr

Description

Generates plots for mlr3::PredictionRegr, depending on argument `type`:

- `"xy"` (default): Scatterplot of "true" response vs. "predicted" response. By default a linear model is fitted via `geom_smooth(method = "lm")` to visualize the trend between x and y (by default colored blue).
  - In addition `geom_abline()` with `slope = 1` is added to the plot.
  - Note that `geom_smooth()` and `geom_abline()` may overlap, depending on the given data.
- `"histogram"`: Histogram of residuals: $r = y - \hat{y}$.
- `"residual"`: Plot of the residuals, with the response $\hat{y}$ on the "x" and the residuals on the "y" axis.
  - By default a linear model is fitted via `geom_smooth(method = "lm")` to visualize the trend between x and y (by default colored blue).

Usage

```
## S3 method for class 'PredictionRegr'
autoplot(object, type = "xy", ...)
```

Arguments

`object`	(mlr3::PredictionRegr).
`type`	(character(1)): Type of the plot. See description.
`...`	(any): Additional arguments, passed down to the respective `geom`.

Value

ggplot2::ggplot() object.

Examples

```
library(mlr3)
library(mlr3viz)

task = tsk("boston_housing")
learner = lrn("regr.rpart")
object = learner$train(task)$predict(task)

head(fortify(object))
autoplot(object)
autoplot(object, type = "histogram", binwidth = 1)
autoplot(object, type = "residual")
```

---

autoplot.PredictionSurv

Plot for PredictionSurv

Description

Generates plots for mlr3proba::PredictionSurv, depending on argument `type`:

- `"calib"` (default): Calibration plot comparing the average predicted survival distribution to a Kaplan-Meier prediction, this is *not* a comparison of a stratified `crank` or `lp` prediction. `object` must have `distr` prediction. `geom_line()` is used for comparison split between the prediction (`Pred`) and Kaplan-Meier estimate (`KM`). In addition labels are added for the x (`T`) and y (`S(T)`) axes.
- `"dcalib"`: Distribution calibration plot. A model is D-calibrated if X% of deaths occur before the X/100 quantile of the predicted distribution, e.g. if 50% of observations die before their predicted median survival time. A model is D-calibrated if the resulting plot lies on x = y.

Usage

```
## S3 method for class 'PredictionSurv'
autoplot(
  object,
  type = c("calib", "dcalib"),
  task = NULL,
  row_ids = NULL,
  times = NULL,
  xyline = TRUE,
  cuts = 11L,
  ...
)
```

Arguments

object	(mlr3proba::PredictionSurv).
type	(character(1)): Type of the plot. See description.
task	(mlr3proba::TaskSurv) If type = "calib" then task is passed to $predict in the Kaplan-Meier learner.
row_ids	(integer()) If type = "calib" then row_ids is passed to $predict in the Kaplan-Meier learner.
times	(numeric()) If type = "calib" then times is the values on the x-axis to plot over, if NULL uses all times from task.
xyline	(logical(1)) If TRUE (default) plots the x-y line for type = "dcalib".
cuts	(integer(1)) Number of cuts in (0,1) to plot dcalib over, default is 11.
...	(any): Additional arguments, currently unused.

References

Haider H, Hoehn B, Davis S, Greiner R (2020). "Effective Ways to Build and Evaluate Individual Survival Distributions." *Journal of Machine Learning Research*, **21**(85), 1-63. https://jmlr.org/papers/v21/18-772.html.

Examples

```
library(mlr3)
library(mlr3proba)
library(mlr3viz)

learn = lrn("surv.coxph")
task = tsk("unemployment")
p = learn$train(task, row_ids = 1:300)$predict(task, row_ids = 301:400)

# calibration by comparison of average prediction to Kaplan-Meier
autoplot(p, type = "calib", task = task, row_ids = 301:400)

# Distribution-calibration (D-Calibration)
autoplot(p, type = "dcalib")
```

---

autoplot.ResampleResult

Plot for ResampleResult

Description

Generates plots for mlr3::ResampleResult, depending on argument `type`:

- `"boxplot"` (default): Boxplot of performance measures.
- `"histogram"`: Histogram of performance measures.
- `"roc"`: ROC curve (1 - specificity on x, sensitivity on y). The predictions of the individual mlr3::Resamplings are merged prior to calculating the ROC curve (micro averaged). Requires package **precrec**. Additional arguments will be passed down to the respective `autoplot()` function in package **precrec**. Arguments `calc_avg` and `cb_alpha` are passed to `precrec::evalmod()`.
- `"prc"`: Precision recall curve. See `"roc"`.
- `"prediction"`: Plots the learner prediction for a grid of points. Needs models to be stored. Set `store_models = TRUE` for [mlr3::resample]. For classification, we support tasks with exactly two features and learners with predict_type= set to `"response"` or `"prob"`. For regression, we support tasks with one or two features. For tasks with one feature we can print confidence bounds if the predict type of the learner was set to `"se"`. For tasks with two features the predict type will be ignored.

Usage

```
## S3 method for class 'ResampleResult'
autoplot(object, type = "boxplot", measure = NULL, predict_sets = "test", ...)
```

Arguments

`object`	(mlr3::ResampleResult).
`type`	(character(1)): Type of the plot. See description.
`measure`	(mlr3::Measure).
`predict_sets`	(`character()`) Only for `type` set to `"prediction"`. Which points should be shown in the plot? Can be a subset of (`"train"`, `"test"`) or empty.
`...`	(any): Additional arguments, passed down to the respective `geom` or plotting function.

Value

`ggplot2::ggplot()` object.

References

Saito T, Rehmsmeier M (2017). "Precrec: fast and accurate precision-recall and ROC curve calculations in R." *Bioinformatics*, **33**(1), 145-147. doi: 10.1093/bioinformatics/btw570.

Examples

```
library(mlr3)
library(mlr3viz)

task = tsk("sonar")
learner = lrn("classif.rpart", predict_type = "prob")
resampling = rsmp("cv")
object = resample(task, learner, resampling)

head(fortify(object))

# Default: boxplot
autoplot(object)

# Histogram
autoplot(object, type = "histogram", bins = 30)

# ROC curve, averaged over resampling folds:
autoplot(object, type = "roc")

# ROC curve of joint prediction object:
autoplot(object$prediction(), type = "roc")

# Precision Recall Curve
autoplot(object, type = "prc")

# Prediction Plot
task = tsk("iris")$select(c("Sepal.Length", "Sepal.Width"))
resampling = rsmp("cv", folds = 3)
object = resample(task, learner, resampling, store_models = TRUE)
autoplot(object, type = "prediction")
```

---

autoplot.TaskClassif *Plot for Classification Tasks*

Description

Generates plots for mlr3::TaskClassif, depending on argument `type`:

- `"target"` (default): Bar plot of the target variable (default).
- `"duo"`: Passes data and additional arguments down to `GGally::ggduo()`. `columnsX` is target, `columnsY` is features.
- `"pairs"`: Passes data and additional arguments down to `GGally::ggpairs()`. Color is set to target column.

Usage

```
## S3 method for class 'TaskClassif'
autoplot(object, type = "target", ...)
```

Arguments

object	(mlr3::TaskClassif).
type	(character(1)): Type of the plot. See description.
...	(any): Additional argument, possibly passed down to the underlying plot functions.

Value

ggplot2::ggplot() object.

Examples

```
library(mlr3)
library(mlr3viz)

task = tsk("iris")

head(fortify(task))
autoplot(task)
autoplot(task$clone()$select(c("Sepal.Length", "Sepal.Width")),
  type = "pairs")
autoplot(task, type = "duo")
```

---

autoplot.TaskClust *Plot for Clustering Tasks*

Description

Generates plots for mlr3cluster::TaskClust, depending on argument type:

- "pairs": Passes data and additional arguments down to GGally::ggpairs() (default).

Usage

```
## S3 method for class 'TaskClust'
autoplot(object, type = "pairs", ...)
```

Arguments

object	(mlr3cluster::TaskClust).
type	(character(1)): Type of the plot. See description.
...	(any): Additional argument, passed down to the underlying geom or plot functions.

Value

ggplot2::ggplot() object.

Examples

```
library(mlr3)
library(mlr3cluster)
library(mlr3viz)

task = mlr_tasks$get("usarrests")

head(fortify(task))
autoplot(task)
```

---

autoplot.TaskDens *Plot for Density Tasks*

Description

Generates plots for mlr3proba::TaskDens.

Usage

```
## S3 method for class 'TaskDens'
autoplot(object, type = "dens", ...)
```

Arguments

object	(mlr3proba::TaskDens).
type	(character(1)): Type of the plot. Available choices: • "dens": histogram density estimator (default) with ggplot2::geom_histogram(). • "freq": histogram frequency plot with ggplot2::geom_histogram(). • "overlay": histogram with overlaid density plot with ggplot2::geom_histogram() and ggplot2::geom_density(). • "freqpoly": frequency polygon plot with ggplot2::geom_freqpoly.
...	(any): Additional arguments, possibly passed down to the underlying plot functions.

Value

ggplot2::ggplot() object.

Examples

```
library(mlr3)
library(mlr3proba)
task = tsk("precip")

head(fortify(task))
autoplot(task, bins = 15)
autoplot(task, type = "freq", bins = 15)
autoplot(task, type = "overlay", bins = 15)
autoplot(task, type = "freqpoly", bins = 15)
```

`autoplot.TaskRegr` *Plot for Regression Tasks*

Description

Generates plots for mlr3::TaskRegr, depending on argument `type`:

- `"target"`: Box plot of target variable (default).
- `"pairs"`: Passes data and additional arguments down to `GGally::ggpairs()`. Color is set to target column.

Usage

```
## S3 method for class 'TaskRegr'
autoplot(object, type = "target", ...)
```

Arguments

`object`	(mlr3::TaskRegr).
`type`	(character(1)): Type of the plot. See description.
`...`	(any): Additional argument, passed down to the underlying `geom` or plot functions.

Value

`ggplot2::ggplot()` object.

Examples

```
library(mlr3)
library(mlr3viz)

task = tsk("mtcars")
task$select(c("am", "carb"))

head(fortify(task))
autoplot(task)
autoplot(task, type = "pairs")
```

`autoplot.TaskSurv` *Plot for Survival Tasks*

Description

Generates plots for mlr3proba::TaskSurv, depending on argument `type`:

- `"target"`: Calls `GGally::ggsurv()` on a `survival::survfit()` object.
- `"duo"`: Passes data and additional arguments down to `GGally::ggduo()`. `columnsX` is target, `columnsY` is features.
- `"pairs"`: Passes data and additional arguments down to `GGally::ggpairs()`. Color is set to target column.

Usage

```
## S3 method for class 'TaskSurv'
autoplot(object, type = "target", ...)
```

Arguments

`object`	(mlr3proba::TaskSurv).
`type`	(`character(1)`): Type of the plot. Available choices:
`...`	(any): Additional argument, passed down to $formula of mlr3proba::TaskSurv or the underlying plot functions.

Value

`ggplot2::ggplot()` object.

Examples

```
library(mlr3)
library(mlr3viz)
library(mlr3proba)

task = tsk("lung")

head(fortify(task))
autoplot(task)
autoplot(task, rhs = "sex")
autoplot(task, type = "duo")
```

`autoplot.TuningInstanceSingleCrit` *Plot for TuningInstanceSingleCrit*

Description

Generates plots for mlr3tuning::TuningInstanceSingleCrit.

Usage

```
## S3 method for class 'TuningInstanceSingleCrit'
autoplot(
  object,
  type = "marginal",
  cols_x = NULL,
  trafo = FALSE,
  learner = mlr3::lrn("regr.ranger"),
  grid_resolution = 100,
  ...
)
```

Arguments

`object`	(mlr3tuning::TuningInstanceSingleCrit.
`type`	(`character(1)`): Type of the plot. Available choices: • `"marginal"`: scatter plots of hyperparameter versus performance. The colour of the points shows the batch number. • `"performance"`: scatter plots of batch number versus performance. • `"parameter"`: scatter plots of batch number versus hyperparameter. The colour of the points shows the performance. • `"parallel"` parallel coordinates plot. Parameter values are rescaled by `(x -mean(x)) / sd(x)`. • `"points"` - scatter plot of two hyperparameters versus performance. The colour of the points shows the performance. • `"surface"`: surface plot of 2 hyperparameters versus performance. The performance values are interpolated with the supplied mlr3::Learner.
`cols_x`	(`character()`) Column names of hyperparameters. By default, all untransformed hyperparameters are plottet. Transformed hyperparameters are prefixed with `x_domain_`.
`trafo`	(`logical(1)`) Determines if untransformed (`FALSE`) or transformed (`TRUE`) hyperparametery are plotted.
`learner`	(mlr3::Learner) Regression learner used to interpolate the data of the surface plot.

grid_resolution
: (numeric())
Resolution of the surface plot.

...
: (any): Additional arguments, possibly passed down to the underlying plot functions.

Value

ggplot2::ggplot() object.

Examples

```
if(requireNamespace("mlr3tuning") && requireNamespace("patchwork")) {
library(mlr3tuning)

learner = lrn("classif.rpart")
learner$param_set$values$cp = to_tune(0.001, 0.1)
learner$param_set$values$minsplit = to_tune(1, 10)

instance = TuningInstanceSingleCrit$new(
  task = tsk("iris"),
  learner = learner,
  resampling = rsmp("holdout"),
  measure = msr("classif.ce"),
  terminator = trm("evals", n_evals = 10))

tuner = tnr("random_search")

tuner$optimize(instance)

# plot performance versus batch number
autoplot(instance, type = "performance")

# plot cp values versus performance
autoplot(instance, type = "marginal", cols_x = "cp")

# plot transformed parameter values versus batch number
autoplot(instance, type = "parameter", trafo = TRUE)

# plot parallel coordinates plot
autoplot(instance, type = "parallel")}
```

---

plot_learner_prediction

Plot for Learner Predictions

Description

Generates a plot for the mlr3::Prediction of a single mlr3::Learner on a single mlr3::Task.

- For classification we support tasks with exactly two features and learners with `predict_type` set to `"response"` or `"prob"`.
- For regression we support tasks with one or two features. For tasks with one feature we print confidence bounds if the predict type of the learner was set to `"se"`. For tasks with two features the predict type will be ignored.

Note that this function is a wrapper around `autoplot.ResampleResult()` for a temporary mlr3::ResampleResult using mlr3::mlr_resamplings_holdout with ratio 1 (all observations in training set).

Usage

```
plot_learner_prediction(learner, task, grid_points = 100L, expand_range = 0)
```

Arguments

`learner`	(mlr3::Learner).
`task`	(mlr3::Task).
`grid_points`	(`integer(1)`) Resolution of the grid. For factors, ordered and logicals this value is ignored.
`expand_range`	(`numeric(1)`) Expand the prediction range for numerical features.

Value

`ggplot2::ggplot()` object.

Examples

```
library(mlr3)
library(mlr3viz)

task = mlr3::tsk("pima")$select(c("age", "glucose"))
learner = lrn("classif.rpart", predict_type = "prob")
p = plot_learner_prediction(learner, task)
print(p)
```

---

predict_grid *Generates a data.table of evenly distributed points.*

Description

For each point we have the predicted class / regression value in column response. If the learner predicts probabilities, a column ".prob.response" is added that contains the probability of the predicted class

Usage

```
predict_grid(learners, task, grid_points, expand_range)
```

Arguments

learners	list of trained learners, each learner belongs to one resampling iteration
task	the task all learners are trained on
grid_points	(int): see sequenize
expand_range	see sequenize

Index