Package 'ROCR'

May 7, 2026

Version 1.0-12

Date 2026-01-22

Title Visualizing the Performance of Scoring Classifiers

Description ROC graphs, sensitivity/specificity curves, lift charts, and precision/recall plots are popular examples of trade-off visualizations for specific pairs of performance measures. ROCR is a flexible tool for creating cutoff-parameterized 2D performance curves by freely combining two from over 25 performance measures (new performance measures can be added using a standard interface). Curves from different cross-validation or bootstrapping runs can be averaged by different methods, and standard deviations, standard errors or box plots can be used to visualize the variability across the runs. The parameterization can be visualized by printing cutoff values at the corresponding curve positions, or by coloring the curve according to cutoff. All components of a performance plot can be quickly adjusted using a flexible parameter dispatching mechanism. Despite its flexibility, ROCR is easy to use, with only three commands and reasonable default values for all optional parameters.

Encoding UTF-8

License GPL (>= 2)

NeedsCompilation no

Depends R (>= 3.6)

Imports methods, graphics, grDevices, gplots, stats

Suggests testthat, knitr, rmarkdown

URL https://ipa-tys.github.io/ROCR/

BugReports https://github.com/ipa-tys/ROCR/issues

RoxygenNote 7.3.3

VignetteBuilder knitr

Author Tobias Sing [aut],
Oliver Sander [aut],
Niko Beerenwinkel [aut],
Thomas Lengauer [aut],
Thomas Unterthiner [ctb],
Felix G.M. Ernst [cre] (ORCID: <https://orcid.org/0000-0001-5064-0928>)

Maintainer Felix G.M. Ernst <felix.gm.ernst@outlook.com>

Repository CRAN

Date/Publication 2026-01-23 06:41:23 UTC

Contents

`performance` *Function to create performance objects*

Description

All kinds of predictor evaluations are performed using this function.

Usage

```
performance(prediction.obj, measure, x.measure = "cutoff", ...)
```

Arguments

`prediction.obj`	An object of class `prediction`.
`measure`	Performance measure to use for the evaluation. A complete list of the performance measures that are available for `measure` and `x.measure` is given in the 'Details' section.
`x.measure`	A second performance measure. If different from the default, a two-dimensional curve, with `x.measure` taken to be the unit in direction of the `x` axis, and `measure` to be the unit in direction of the `y` axis, is created. This curve is parametrized with the cutoff.
`...`	Optional arguments (specific to individual performance measures).

Details

Here is the list of available performance measures. Let Y and $\hat{Y}$ be random variables representing the class and the prediction for a randomly drawn sample, respectively. We denote by $\oplus$ and $\ominus$ the positive and negative class, respectively. Further, we use the following abbreviations for empirical quantities: P (# positive samples), N (# negative samples), TP (# true positives), TN (# true negatives), FP (# false positives), FN (# false negatives).

`acc`: Accuracy. $P(\hat{Y} = Y)$. Estimated as: $\frac{TP+TN}{P+N}$.

`err`: Error rate. $P(\hat{Y} \neq Y)$. Estimated as: $\frac{FP+FN}{P+N}$.

`fpr`: False positive rate. $P(\hat{Y} = \oplus | Y = \ominus)$. Estimated as: $\frac{FP}{N}$.

`fall`: Fallout. Same as `fpr`.

`tpr`: True positive rate. $P(\hat{Y} = \oplus | Y = \oplus)$. Estimated as: $\frac{TP}{P}$.

`rec`: Recall. Same as `tpr`.

`sens`: Sensitivity. Same as `tpr`.

`fnr`: False negative rate. $P(\hat{Y} = \ominus | Y = \oplus)$. Estimated as: $\frac{FN}{P}$.

`miss`: Miss. Same as `fnr`.

`tnr`: True negative rate. $P(\hat{Y} = \ominus | Y = \ominus)$.

`spec`: Specificity. Same as `tnr`.

`ppv`: Positive predictive value. $P(Y = \oplus | \hat{Y} = \oplus)$. Estimated as: $\frac{TP}{TP+FP}$.

`prec`: Precision. Same as `ppv`.

`npv`: Negative predictive value. $P(Y = \ominus | \hat{Y} = \ominus)$. Estimated as: $\frac{TN}{TN+FN}$.

`pcfall`: Prediction-conditioned fallout. $P(Y = \ominus | \hat{Y} = \oplus)$. Estimated as: $\frac{FP}{TP+FP}$.

`pcmiss`: Prediction-conditioned miss. $P(Y = \oplus | \hat{Y} = \ominus)$. Estimated as: $\frac{FN}{TN+FN}$.

`rpp`: Rate of positive predictions. $P(\hat{Y} = \oplus)$. Estimated as: (TP+FP)/(TP+FP+TN+FN).

`rnp`: Rate of negative predictions. $P(\hat{Y} = \ominus)$. Estimated as: (TN+FN)/(TP+FP+TN+FN).

`phi`: Phi correlation coefficient. $\frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FN) \cdot (TN+FP) \cdot (TP+FP) \cdot (TN+FN)}}$. Yields a number between -1 and 1, with 1 indicating a perfect prediction, 0 indicating a random prediction. Values below 0 indicate a worse than random prediction.

`mat`: Matthews correlation coefficient. Same as `phi`.

`mi`: Mutual information. $I(\hat{Y}, Y) := H(Y) - H(Y|\hat{Y})$, where H is the (conditional) entropy. Entropies are estimated naively (no bias correction).

`chisq`: Chi square test statistic. `?chisq.test` for details. Note that R might raise a warning if the sample size is too small.

`odds`: Odds ratio. $\frac{TP \cdot TN}{FN \cdot FP}$. Note that odds ratio produces Inf or NA values for all cutoffs corresponding to FN=0 or FP=0. This can substantially decrease the plotted cutoff region.

`lift`: Lift value. $\frac{P(\hat{Y}=\oplus | Y=\oplus)}{P(\hat{Y}=\oplus)}$.

`f`: Precision-recall F measure (van Rijsbergen, 1979). Weighted harmonic mean of precision (P) and recall (R). $F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}}$. If $\alpha = \frac{1}{2}$, the mean is balanced. A frequent equivalent formulation is $F = \frac{(\beta^2+1) \cdot P \cdot R}{R + \beta^2 \cdot P}$. In this formulation, the mean is balanced if $\beta = 1$. Currently, ROCR only accepts the alpha version as input (e.g. $\alpha = 0.5$). If no value for alpha is given, the mean will be balanced by default.

`rch`: ROC convex hull. A ROC (=`tpr` vs `fpr`) curve with concavities (which represent suboptimal choices of cutoff) removed (Fawcett 2001). Since the result is already a parametric performance curve, it cannot be used in combination with other measures.

`auc`: Area under the ROC curve. This is equal to the value of the Wilcoxon-Mann-Whitney test statistic and also the probability that the classifier will score are randomly drawn positive sample higher than a randomly drawn negative sample. Since the output of `auc` is cutoff-independent, this measure cannot be combined with other measures into a parametric curve. The partial area under the ROC curve up to a given false positive rate can be calculated by passing the optional parameter `fpr.stop=0.5` (or any other value between 0 and 1) to `performance`.

`aucpr`: Area under the Precision/Recall curve. Since the output of `aucpr` is cutoff-independent, this measure cannot be combined with other measures into a parametric curve.

`prbe`: Precision-recall break-even point. The cutoff(s) where precision and recall are equal. At this point, positive and negative predictions are made at the same rate as their prevalence in the data. Since the output of `prbe` is just a cutoff-independent scalar, this measure cannot be combined with other measures into a parametric curve.

`cal`: Calibration error. The calibration error is the absolute difference between predicted confidence and actual reliability. This error is estimated at all cutoffs by sliding a window across the range of possible cutoffs. The default window size of 100 can be adjusted by passing the optional parameter `window.size=200` to `performance`. E.g., if for several positive samples the output of the classifier is around 0.75, you might expect from a well-calibrated classifier that the fraction of them which is correctly predicted as positive is also around 0.75. In a well-calibrated classifier, the probabilistic confidence estimates are realistic. Only for use with probabilistic output (i.e. scores between 0 and 1).

`mxe`: Mean cross-entropy. Only for use with probabilistic output. $MXE := -\frac{1}{P+N}(\sum_{y_i=\oplus} ln(\hat{y}_i) + \sum_{y_i=\ominus} ln(1-\hat{y}_i))$. Since the output of `mxe` is just a cutoff-independent scalar, this measure cannot be combined with other measures into a parametric curve.

`rmse`: Root-mean-squared error. Only for use with numerical class labels. $RMSE := \sqrt{\frac{1}{P+N}\sum_i (y_i - \hat{y}_i)^2}$. Since the output of `rmse` is just a cutoff-independent scalar, this measure cannot be combined with other measures into a parametric curve.

`sar`: Score combinining performance measures of different characteristics, in the attempt of creating a more "robust" measure (cf. Caruana R., ROCAI2004): SAR = 1/3 * (Accuracy + Area under the ROC curve + Root mean-squared error).

`ecost`: Expected cost. For details on cost curves, cf. Drummond&Holte 2000,2004. `ecost` has an obligatory x axis, the so-called 'probability-cost function'; thus it cannot be combined with other measures. While using `ecost` one is interested in the lower envelope of a set of lines, it might be instructive to plot the whole set of lines in addition to the lower envelope. An example is given in `demo(ROCR)`.

`cost`: Cost of a classifier when class-conditional misclassification costs are explicitly given. Accepts the optional parameters `cost.fp` and `cost.fn`, by which the costs for false positives and negatives can be adjusted, respectively. By default, both are set to 1.

Value

An S4 object of class `performance`.

Note

Here is how to call `performance()` to create some standard evaluation plots:

ROC curves: measure="tpr", x.measure="fpr".

Precision/recall graphs: measure="prec", x.measure="rec".

Sensitivity/specificity plots: measure="sens", x.measure="spec".

Lift charts: measure="lift", x.measure="rpp".

Author(s)

Tobias Sing <tobias.sing@gmail.com>, Oliver Sander <osander@gmail.com>

References

A detailed list of references can be found on the ROCR homepage at https://ipa-tys.github.io/ROCR/.

See Also

prediction, prediction-class, performance-class, plot.performance

Examples

```
# computing a simple ROC curve (x-axis: fpr, y-axis: tpr)
library(ROCR)
data(ROCR.simple)
pred <- prediction( ROCR.simple$predictions, ROCR.simple$labels)
pred
perf <- performance(pred,"tpr","fpr")
perf
plot(perf)

# precision/recall curve (x-axis: recall, y-axis: precision)
perf <- performance(pred, "prec", "rec")
perf
plot(perf)

# sensitivity/specificity curve (x-axis: specificity,
# y-axis: sensitivity)
perf <- performance(pred, "sens", "spec")
perf
plot(perf)
```

performance-class *Class* performance

Description

Object to capture the result of a performance evaluation, optionally collecting evaluations from several cross-validation or bootstrapping runs.

Details

A performance object can capture information from four different evaluation scenarios:

- The behaviour of a cutoff-dependent performance measure across the range of all cutoffs (e.g. performance(predObj, 'acc')). Here, x.values contains the cutoffs, y.values the corresponding values of the performance measure, and alpha.values is empty.

- The trade-off between two performance measures across the range of all cutoffs (e.g. performance(predObj, 'tpr', 'fpr')). In this case, the cutoffs are stored in alpha.values, while x.values and y.values contain the corresponding values of the two performance measures.

- A performance measure that comes along with an obligatory second axis (e.g. performance(predObj, 'ecost')). Here, the measure values are stored in y.values, while the corresponding values of the obligatory axis are stored in x.values, and alpha.values is empty.

- A performance measure whose value is just a scalar (e.g. performance(predObj, 'auc')). The value is then stored in y.values, while x.values and alpha.values are empty.

Slots

x.name Performance measure used for the x axis.

y.name Performance measure used for the y axis.

alpha.name Name of the unit that is used to create the parametrized curve. Currently, curves can only be parametrized by cutoff, so alpha.name is either none or cutoff.

x.values A list in which each entry contains the x values of the curve of this particular cross-validation run. x.values[[i]], y.values[[i]], and alpha.values[[i]] correspond to each other.

y.values A list in which each entry contains the y values of the curve of this particular cross-validation run.

alpha.values A list in which each entry contains the cutoff values of the curve of this particular cross-validation run.

Objects from the Class

Objects can be created by using the performance function.

Author(s)

Tobias Sing <tobias.sing@gmail.com>, Oliver Sander <osander@gmail.com>

References

A detailed list of references can be found on the ROCR homepage at https://ipa-tys.github.io/ROCR/.

See Also

prediction performance, prediction-class, plot.performance

---

plot-methods *Plot method for performance objects*

Description

This is the method to plot all objects of class performance.

Usage

```
## S4 method for signature 'performance,missing'
plot(
  x,
  y,
  ...,
  avg = "none",
  spread.estimate = "none",
  spread.scale = 1,
  show.spread.at = c(),
  colorize = FALSE,
  colorize.palette = rev(rainbow(256, start = 0, end = 4/6)),
  colorkey = colorize,
  colorkey.relwidth = 0.25,
  colorkey.pos = "right",
  print.cutoffs.at = c(),
  cutoff.label.function = function(x) {
     round(x, 2)
 },
  downsampling = 0,
  add = FALSE
)

## S3 method for class 'performance'
plot(...)
```

Arguments

x	an object of class performance
y	not used
...	Optional graphical parameters to adjust different components of the performance plot. Parameters are directed to their target component by prefixing them with the name of the component (component.parameter, e.g. text.cex). The following components are available: xaxis, yaxis, coloraxis, box (around the plotting region), points, text, plotCI (error bars), boxplot. The names of these components are influenced by the R functions that are used to create them. Thus, par(component) can be used to see which parameters are available for a given component (with the expection of the three axes; use par(axis) here). To adjust the canvas or the performance curve(s), the standard plot parameters can be used without any prefix.
avg	If the performance object describes several curves (from cross-validation runs or bootstrap evaluations of one particular method), the curves from each of the runs can be averaged. Allowed values are none (plot all curves separately), horizontal (horizontal averaging), vertical (vertical averaging), and threshold (threshold (=cutoff) averaging). Note that while threshold averaging is always feasible, vertical and horizontal averaging are not well-defined if the graph cannot be represented as a function x->y and y->x, respectively.
spread.estimate	When curve averaging is enabled, the variation around the average curve can be visualized as standard error bars (stderror), standard deviation bars (stddev), or by using box plots (boxplot). Note that the function plotCI, which is used internally by ROCR to draw error bars, might raise a warning if the spread of the curves at certain positions is 0.
spread.scale	For stderror or stddev, this is a scalar factor to be multiplied with the length of the standard error/deviation bar. For example, under normal assumptions, spread.scale=2 can be used to get approximate 95% confidence intervals.
show.spread.at	For vertical averaging, this vector determines the x positions for which the spread estimates should be visualized. In contrast, for horizontal and threshold averaging, the y positions and cutoffs are determined, respectively. By default, spread estimates are shown at 11 equally spaced positions.
colorize	This logical determines whether the curve(s) should be colorized according to cutoff.
colorize.palette	If curve colorizing is enabled, this determines the color palette onto which the cutoff range is mapped.
colorkey	If true, a color key is drawn into the 4% border region (default of par(xaxs) and par(yaxs)) of the plot. The color key visualizes the mapping from cutoffs to colors.
colorkey.relwidth	Scalar between 0 and 1 that determines the fraction of the 4% border region that is occupied by the colorkey.
colorkey.pos	Determines if the colorkey is drawn vertically at the right side, or horizontally at the top of the plot.

print.cutoffs.at
: This vector specifies the cutoffs which should be printed as text along the curve at the corresponding curve positions.

cutoff.label.function
: By default, cutoff annotations along the curve or at the color key are rounded to two decimal places before printing. Using a custom `cutoff.label.function`, any other transformation can be performed on the cutoffs instead (e.g. rounding with different precision or taking the logarithm).

downsampling
: ROCR can efficiently compute most performance measures even for data sets with millions of elements. However, plotting of large data sets can be slow and lead to PS/PDF documents of considerable size. In that case, performance curves that are indistinguishable from the original can be obtained by using only a fraction of the computed performance values. Values for downsampling between 0 and 1 indicate the fraction of the original data set size to which the performance object should be downsampled, integers above 1 are interpreted as the actual number of performance values to which the curve(s) should be downsampled.

add
: If `TRUE`, the curve(s) is/are added to an already existing plot; otherwise a new plot is drawn.

Author(s)

Tobias Sing <tobias.sing@gmail.com>, Oliver Sander <osander@gmail.com>

References

A detailed list of references can be found on the ROCR homepage at https://ipa-tys.github.io/ROCR/.

See Also

prediction, performance, prediction-class, performance-class

Examples

```
# plotting a ROC curve:
library(ROCR)
data(ROCR.simple)
pred <- prediction( ROCR.simple$predictions, ROCR.simple$labels )
pred
perf <- performance( pred, "tpr", "fpr" )
perf
plot( perf )

# To entertain your children, make your plots nicer
# using ROCR's flexible parameter passing mechanisms
# (much cheaper than a finger painting set)
par(bg="lightblue", mai=c(1.2,1.5,1,1))
plot(perf, main="ROCR fingerpainting toolkit", colorize=TRUE,
     xlab="Mary's axis", ylab="", box.lty=7, box.lwd=5,
```

```
box.col="gold", lwd=17, colorkey.relwidth=0.5, xaxis.cex.axis=2,
xaxis.col='blue', xaxis.col.axis="blue", yaxis.col='green', yaxis.cex.axis=2,
yaxis.at=c(0,0.5,0.8,0.85,0.9,1), yaxis.las=1, xaxis.lwd=2, yaxis.lwd=3,
yaxis.col.axis="orange", cex.lab=2, cex.main=2)
```

`prediction` *Function to create prediction objects*

Description

Every classifier evaluation using ROCR starts with creating a `prediction` object. This function is used to transform the input data (which can be in vector, matrix, data frame, or list form) into a standardized format.

Usage

```
prediction(predictions, labels, label.ordering = NULL)
```

Arguments

`predictions`	A vector, matrix, list, or data frame containing the predictions.
`labels`	A vector, matrix, list, or data frame containing the true class labels. Must have the same dimensions as `predictions`.
`label.ordering`	The default ordering (cf.details) of the classes can be changed by supplying a vector containing the negative and the positive class label.

Details

`predictions` and `labels` can simply be vectors of the same length. However, in the case of cross-validation data, different cross-validation runs can be provided as the *columns* of a matrix or data frame, or as the entries of a list. In the case of a matrix or data frame, all cross-validation runs must have the same length, whereas in the case of a list, the lengths can vary across the cross-validation runs. Internally, as described in section 'Value', all of these input formats are converted to list representation.

Since scoring classifiers give relative tendencies towards a negative (low scores) or positive (high scores) class, it has to be declared which class label denotes the negative, and which the positive class. Ideally, labels should be supplied as ordered factor(s), the lower level corresponding to the negative class, the upper level to the positive class. If the labels are factors (unordered), numeric, logical or characters, ordering of the labels is inferred from R's built-in < relation (e.g. 0 < 1, -1 < 1, 'a' < 'b', FALSE < TRUE). Use `label.ordering` to override this default ordering. Please note that the ordering can be locale-dependent e.g. for character labels '-1' and '1'.

Currently, ROCR supports only binary classification (extensions toward multiclass classification are scheduled for the next release, however). If there are more than two distinct label symbols, execution stops with an error message. If all predictions use the same two symbols that are used for the labels, categorical predictions are assumed. If there are more than two predicted values, but all numeric, continuous predictions are assumed (i.e. a scoring classifier). Otherwise, if more than two symbols occur in the predictions, and not all of them are numeric, execution stops with an error message.

Value

An S4 object of class `prediction`.

Author(s)

Tobias Sing <tobias.sing@gmail.com>, Oliver Sander <osander@gmail.com>

References

A detailed list of references can be found on the ROCR homepage at https://ipa-tys.github.io/ROCR/.

See Also

`prediction-class`, `performance`, `performance-class`, `plot.performance`

Examples

```
# create a simple prediction object
library(ROCR)
data(ROCR.simple)
pred <- prediction(ROCR.simple$predictions,ROCR.simple$labels)
pred
```

---

prediction-class *Class* prediction

Description

Object to encapsulate numerical predictions together with the corresponding true class labels, optionally collecting predictions and labels for several cross-validation or bootstrapping runs.

Slots

`predictions` A list, in which each element is a vector of predictions (the list has length > 1 for x-validation data.

`labels` Analogously, a list in which each element is a vector of true class labels.

`cutoffs` A list in which each element is a vector of all necessary cutoffs. Each cutoff vector consists of the predicted scores (duplicates removed), in descending order.

`fp` A list in which each element is a vector of the number (not the rate!) of false positives induced by the cutoffs given in the corresponding 'cutoffs' list entry.

`tp` As fp, but for true positives.

`tn` As fp, but for true negatives.

`fn` As fp, but for false negatives.

`n.pos` A list in which each element contains the number of positive samples in the given x-validation run.

`n.neg` As n.pos, but for negative samples.

`n.pos.pred` A list in which each element is a vector of the number of samples predicted as positive at the cutoffs given in the corresponding 'cutoffs' entry.

`n.neg.pred` As n.pos.pred, but for negatively predicted samples.

Objects from the Class

Objects can be created by using the `prediction` function.

Note

Every `prediction` object contains information about the 2x2 contingency table consisting of tp,tn,fp, and fn, along with the marginal sums n.pos,n.neg,n.pos.pred,n.neg.pred, because these form the basis for many derived performance measures.

Author(s)

Tobias Sing <tobias.sing@gmail.com>, Oliver Sander <osander@gmail.com>

References

A detailed list of references can be found on the ROCR homepage at https://ipa-tys.github.io/ROCR/.

See Also

`prediction`, `performance`, `performance-class`, `plot.performance`

ROCR.hiv *Data set: Support vector machines and neural networks applied to the prediction of HIV-1 coreceptor usage*

Description

Linear support vector machines (libsvm) and neural networks (R package nnet) were applied to predict usage of the coreceptors CCR5 and CXCR4 based on sequence data of the third variable loop of the HIV envelope protein.

Usage

```
data(ROCR.hiv)
```

Format

A list consisting of the SVM (`ROCR.hiv$hiv.svm`) and NN (`ROCR.hiv$hiv.nn`) classification data. Each of those is in turn a list consisting of the two elements `$predictions` and `$labels` (10 element list representing cross-validation data).

References

Sing, T. & Beerenwinkel, N. & Lengauer, T. "Learning mixtures of localized rules by maximizing the area under the ROC curve". 1st International Workshop on ROC Analysis in AI, 89-96, 2004.

Examples

```
library(ROCR)
data(ROCR.hiv)
attach(ROCR.hiv)
pred.svm <- prediction(hiv.svm$predictions, hiv.svm$labels)
pred.svm
perf.svm <- performance(pred.svm, 'tpr', 'fpr')
perf.svm
pred.nn <- prediction(hiv.nn$predictions, hiv.svm$labels)
pred.nn
perf.nn <- performance(pred.nn, 'tpr', 'fpr')
perf.nn
plot(perf.svm, lty=3, col="red",main="SVMs and NNs for prediction of
HIV-1 coreceptor usage")
plot(perf.nn, lty=3, col="blue",add=TRUE)
plot(perf.svm, avg="vertical", lwd=3, col="red",
     spread.estimate="stderror",plotCI.lwd=2,add=TRUE)
plot(perf.nn, avg="vertical", lwd=3, col="blue",
     spread.estimate="stderror",plotCI.lwd=2,add=TRUE)
legend(0.6,0.6,c('SVM','NN'),col=c('red','blue'),lwd=3)
```

ROCR.simple *Data set: Simple artificial prediction data for use with ROCR*

Description

A mock data set containing a simple set of predictions and corresponding class labels.

Usage

```
data(ROCR.simple)
```

Format

A two element list. The first element, `ROCR.simple$predictions`, is a vector of numerical predictions. The second element, `ROCR.simple$labels`, is a vector of corresponding class labels.

Examples

```
# plot a ROC curve for a single prediction run
# and color the curve according to cutoff.
library(ROCR)
data(ROCR.simple)
pred <- prediction(ROCR.simple$predictions, ROCR.simple$labels)
```

```
pred
perf <- performance(pred,"tpr","fpr")
perf
plot(perf,colorize=TRUE)
```

ROCR.xval *Data set: Artificial cross-validation data for use with ROCR*

Description

A mock data set containing 10 sets of predictions and corresponding labels as would be obtained from 10-fold cross-validation.

Usage

```
data(ROCR.xval)
```

Format

A two element list. The first element, `ROCR.xval$predictions`, is itself a 10 element list. Each of these 10 elements is a vector of numerical predictions for each cross-validation run. Likewise, the second list entry, `ROCR.xval$labels` is a 10 element list in which each element is a vector of true class labels corresponding to the predictions.

Examples

```
# plot ROC curves for several cross-validation runs (dotted
# in grey), overlaid by the vertical average curve and boxplots
# showing the vertical spread around the average.
library(ROCR)
data(ROCR.xval)
pred <- prediction(ROCR.xval$predictions, ROCR.xval$labels)
pred
perf <- performance(pred,"tpr","fpr")
perf
plot(perf,col="grey82",lty=3)
plot(perf,lwd=3,avg="vertical",spread.estimate="boxplot",add=TRUE)
```

Index