MLeval

Christopher R John

2020-02-11

MLeval is a machine learning model evaluation package for R. It works with models that yield probabilities for binary labels and evaluates those together with the ground truth probabilities in order to assess performance. MLeval provides a number of metrics suitable for balanced and imbalanced data. MLeval can automatically interpret the results of the Caret train function that performs repeated cross validation and is intended to speed up and simplify analyses performed in this way. It can evaluate probabilities produced by other methods, e.g. a single split into training and test, simply requiring the results manifest as a data frame of ground truth probabilities and predicted probabilities for each class. MLeval contains ggplot2 based functions for producing ROC curves, precision recall curves, precision recall gain curves, and calibration curves.

Contents

1. Introduction

Evaluation is a key part of machine learning and evaluation metrics are subject to bias, so it is best to proceed with some understanding of these biases. Many types of model yield probabilities of classes and one approach is to evaluate these directly with the ground truth probabilities (0 or 1) to calculate 'proper scores'. We will deal with this later in the vignette, however, for data interpretation, it typically comes down to whether an observation is in one class or another and whether that prediction is right or wrong. To achieve this, a probability has to be turned into a classification.

One important fact to remember is 'improper scores' (such as accuracy, F1, kappa, etc) depend on setting a probability cut-off, e.g. probability positive > 0.5 means positive and probability positive < 0.5 means negative. For balanced data this is fine, but in the imbalanced case the positive (minority) class probabilities may be all below 0.5, but the model may still be capable of perfect discrimination. A solution to this is to use 'semi-proper scores' such as the ROC-AUC and PR-AUC that consider the ranking of positives and negative according to probability positive. Let us consider the binary classification task where the aim is to correctly predict the identity of the real positives and negatives.

RP (real positives), RN (real negatives), PP (predicted positives), PN (predicted negatives)
TP (true positives), TN (true negatives), FP (false positives), FN (true negatives)

These definitions have the units of numbers of observations. Note that, predicted means predicted from our particular model and that TP+TN+FP+FN=RP+RN and TP+TN+FP+FN=PP+PN. We also use N as the total number of observations.

Accuracy=TP+TN/TP+TN+FP+FN
TPR=TP/TP+FN (sensitivity/ recall)

TNR=TN/TN+FP (specificity)
PPV=TP/TP+FP (precision/ positive predictive value)
NPV=TN/TN+FN (negative predictive value)
Informedness=TPR+TNR-1
Markedness=PPV+NPV-1

In a medical context, TPR is used to see how many positives are correctly picked up while TNR is used to see how many of the negatives are correctly picked up. Some diseases have a prevalence (probability of disease) of only one in a million, thus the real positives (RP) are very low in number compared with real negatives (RP). If a model always predicts negative you have an Accuracy of 0.999999. This is achieved by the simple ZeroR learner that always predicts the maximum class. If we consider the Recall and Precision for predicting that you are disease free, then we have Recall=1 and Precision=0.999999 for ZeroR, thus the F1 score is near 1. If you reverse positive and negative and try to predict that a person has the disease with ZeroR you get Recall=0 and Precision=0.

So accuracy and the F1 score can be easily shown to be bias metrics that must be carefully interpreted, in fact, there is seldom a good case for using accuracy, so MLeval does not calculate it. While the F1 score is often useful for needle in the haystack kind of problems, when the minority class is set as positive, its value can change considerably depending on which class is defined as positive which can be a problem. For a good discussion of the problems of the F1 score see a recent paper by Powers (2015).

An alternative metric is informedness (Powers, 2011), which in the case of the ZeroR example, would yield 0 which correctly corresponds to the baseline chance expectation. Informedness considers both positive and negative classes. However, on very imbalanced data similar to the ROC-AUC, it can be insensitive to FP's in the positive fraction relative to the F1 because it depends on sensitivity and specificity only. Markedness is the opposite of informedness because it is derived from positive and negative predictive value scores. A good metric that combines them both is Matthew's correlation coefficient that is included in this package.

We recommend, when investigating the discriminative power of a model, to use a semi-proper score after examining a curve, such as the ROC or PR curve. In either case, we explore the system by changing the probability positive cut-off parameter then asking how metrics behave for every threshold, in the case of the ROC we plot TPR vs FPR, for a PR curve it is precision vs recall. Note, the ROC has many preferable statistical properties to the PR curve, such as: the chance line expectation being on the diagnonal, results not being dependent on which class is defined as positive, and a probabilistic interpretation. Yet the PR curve is better suited to extreme class imbalance when we care only about the positive class. We have also implemented the PR gain curve that solves some of the shortcomings of the PR curve, but still is best kept for extreme class imbalances (Flach and Meelis, 2015).

General recommendations for testing models: In the absence of an external validation dataset, we generally recommend 50-100 times repeated 5 or 10 fold cross validation to reduce the bias and variance of the estimations. Prior feature selection should be unbias, meaning not based on a statistical test applied to the entire data (overfitting), i.e. based on variance, cross validation embedded statistical test, or a priori knowledge from other studies. Many models perform some kind of internal feature selection already so adding additional supervised feature selection inside the cross validation loop is not usually helpful.

2. Standard operation: single group

When we split into training and testing datasets once this is the most basic way to evaluate a model. Unless the point size is very high, this approach is prone to bias as each time we split the data there is a degree of randomness in the results. This is why we do repeated cross validation because it reduces this bias and generates better estimates of model performance.

The MLeval 'evalm' function is run on the data frame of ground truth labels and probabilties. It goes as follows, column 1: G1 probabilities, column 2: G2 probabilities, column 3: ground truth labels named 'obs'. Column 4 is an optional column that should be named 'Group' that distinguishes results from different models bound together (see next example). G1 and G2 columns should be named according to their class.

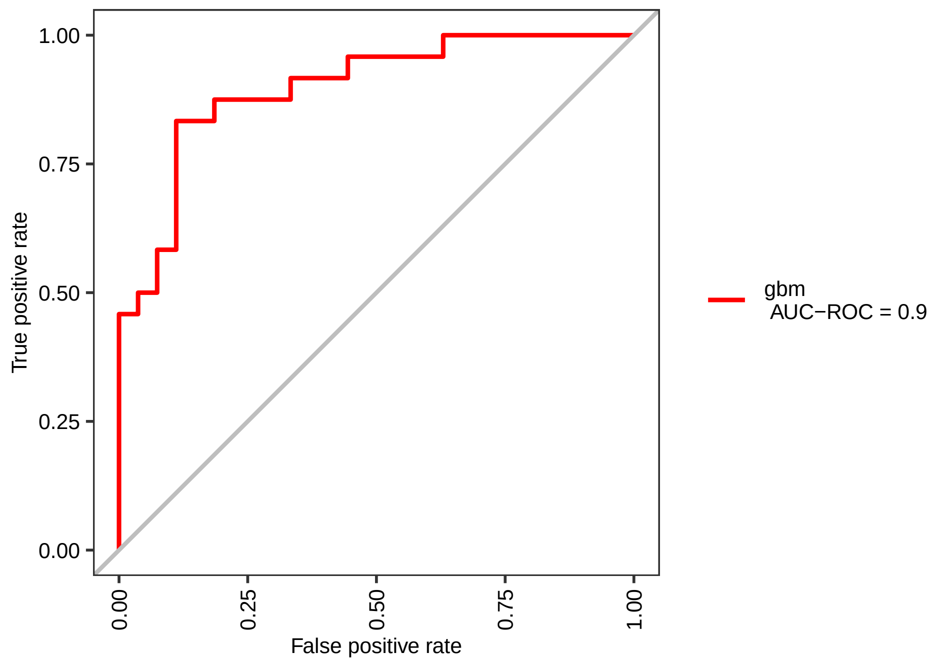
Type ?eval for further information on this function.

We are going to increase the 'bins' for the calibration curve because there are quite a few points.

```
library(MLeval)

## ordinary input
head(preds)
#>           M         R obs Group
#> 1 0.4591177 0.5408823   R   gbm
#> 2 0.8769442 0.1230558   R   gbm
#> 3 0.4608720 0.5391280   R   gbm
#> 4 0.6600493 0.3399507   R   gbm
#> 5 0.5451453 0.4548547   R   gbm
#> 6 0.5002910 0.4997090   R   gbm

## run MLeval
test1 <- evalm(preds,plots='r',rlinethick=0.8,fsize=8,bins=8)
#> ***MLeval: Machine Learning Model Evaluation***
#> Input: data frame of probabilities of observed labels
#> Group column exists.
#> Observations: 51
#> Number of groups: 1
#> Observations per group: 51
#> Positive: R
#> Negative: M
#> Group: gbm
#> Positive: 24
#> Negative: 27
#> ***Performance Metrics***
#> gbm Optimal Informedness = 0.722222222222222
#> gbm AUC-ROC = 0.9
```



The ROC represents an ensemble of operating points corresponding to different probability cut-offs for defining a positive and negative. We can then get other metrics out using the typical p=0.5 cut-off as follows. The results also contain 95% confidence intervals. The ROC-AUC and the PR-AUC are not subject to this cut-off threshold, and are generally preferable to relying on other metrics. Where the ROC-AUC represents a more objective and balanced view because it considers both classes.

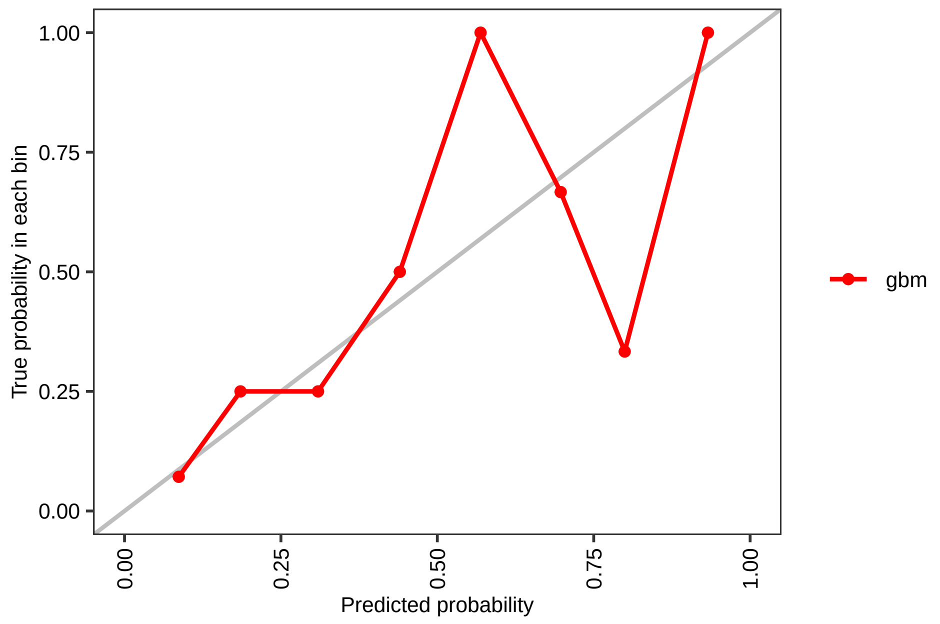
```
test1$stdres
#> $gbm
#>                 Score        CI
#> SENS            0.750 0.55-0.88
#> SPEC            0.889 0.72-0.96
#> MCC             0.648      <NA>
#> Informedness    0.639      <NA>
#> PREC            0.857 0.65-0.95
#> NPV             0.800  0.63-0.9
#> FPR             0.111      <NA>
#> F1              0.800      <NA>
#> TP             18.000      <NA>
#> FP              3.000      <NA>
#> TN             24.000      <NA>
#> FN              6.000      <NA>
#> AUC-ROC         0.900 0.81-0.99
#> AUC-PR          0.850      <NA>
#> AUC-PRG         0.470      <NA>
```

Whereas if we want to extract the results corresponding to the operating point with the maximal informedness we use this command. By default, MLeval uses informedness to select the ROC working point, but this may be customised. This can help especially with imbalanced data, where the p=0.5 cut-off does not often make good sense. There is nothing magical about the p=0.5 cut-off.

```
test1$optres
#> $gbm
#>                 Score        CI
#> SENS            0.833 0.64-0.93
#> SPEC            0.889 0.72-0.96
#> MCC             0.724      <NA>
#> Informedness    0.722      <NA>
#> PREC            0.870 0.68-0.95
#> NPV             0.857 0.69-0.94
#> FPR             0.111      <NA>
#> F1              0.851      <NA>
#> TP             20.000      <NA>
#> FP              3.000      <NA>
#> TN             24.000      <NA>
#> FN              4.000      <NA>
#> AUC-ROC         0.900 0.81-0.99
#> AUC-PR          0.850      <NA>
#> AUC-PRG         0.470      <NA>
```

A calibration curve which compares the predicted probabilities placed into bins with the actual probabilities (ground truth labels) can be retrieved as follows.

```
test1$cc
```



As we can see the calibration curve is quite similar to the idealised situation (diagonal line), therefore the model is fairly well calibrated. Calibration is an important part of determining whether a model produces sensible probability estimates, which is distinct from discrimination, i.e. ROC curve analysis.

It is also possible to extract all probabilities and also all metrics for each corresponding threshold from the results object for further analysis if necessary. If we run on multiple groups there will be a data frame of probabilities for each group in a list, i.e. different models, so we index the list to select the group of interest.

```
head(test1$probs[[1]])
#>              M          R obs Group TP TN FP FN       SENS SPEC Informedness
#> 18 0.02441351 0.9755865   R   gbm  1 27  0 23 0.04166667    1   0.04166667
#> 19 0.03028840 0.9697116   R   gbm  2 27  0 22 0.08333333    1   0.08333333
#> 17 0.04132870 0.9586713   R   gbm  3 27  0 21 0.12500000    1   0.12500000
#> 20 0.04325991 0.9567401   R   gbm  4 27  0 20 0.16666667    1   0.16666667
#> 8  0.05134618 0.9486538   R   gbm  5 27  0 19 0.20833333    1   0.20833333
#> 12 0.05429690 0.9457031   R   gbm  6 27  0 18 0.25000000    1   0.25000000
#>    PREC       NPV      MARK        F1       MCC FPR PG RG
#> 18    1 0.5400000 0.5400000 0.0800000 0.1500000   0  1  0
#> 19    1 0.5510204 0.5510204 0.1538462 0.2142857   0  1  0
#> 17    1 0.5625000 0.5625000 0.2222222 0.2651650   0  1  0
#> 20    1 0.5744681 0.5744681 0.2857143 0.3094264   0  1  0
#> 8     1 0.5869565 0.5869565 0.3448276 0.3496893   0  1  0
#> 12    1 0.6000000 0.6000000 0.4000000 0.3872983   0  1  0
```

3. Standard operation: multiple groups

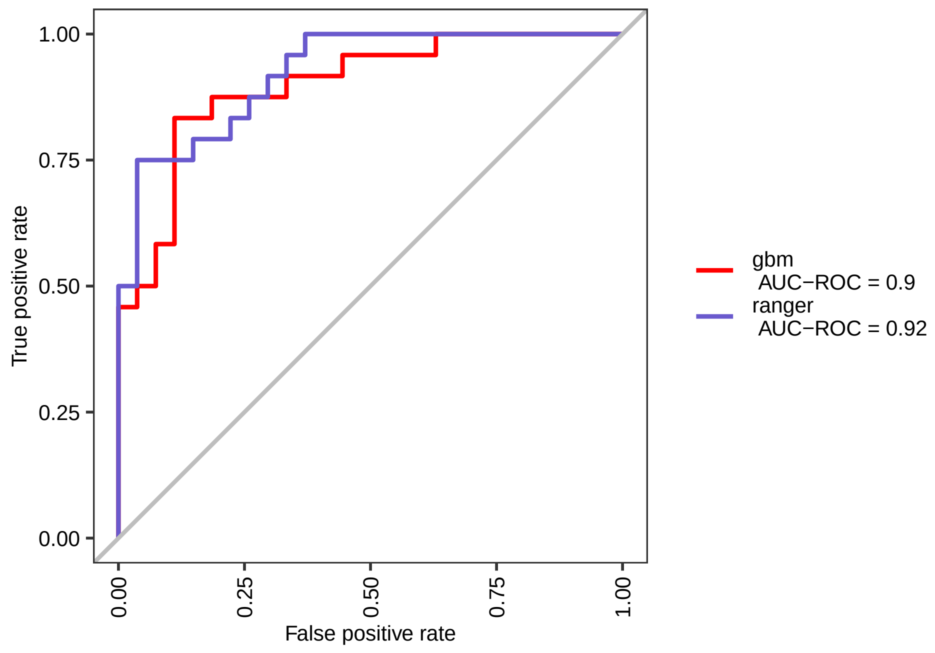
In this example, predictions from different models and real probabilities have been combined horizontally. These can be analysed and plotted together by passing in the combined data, it requires labelling with a ‘Group’ column to specify the different models. Here we are comparing random forest and gbm.

```
library(MLeval)

##
levels(as.factor(predsc$Group))
```

```
#> [1] "gbm"    "ranger"
head(predsc)
#>           M         R obs Group
#> 1 0.4591177 0.5408823   R   gbm
#> 2 0.8769442 0.1230558   R   gbm
#> 3 0.4608720 0.5391280   R   gbm
#> 4 0.6600493 0.3399507   R   gbm
#> 5 0.5451453 0.4548547   R   gbm
#> 6 0.5002910 0.4997090   R   gbm

## run MLeval
test1 <- evalm(predsc,plots='r',rlinethick=0.8,fsize=8,bins=8)
#> ***MLeval: Machine Learning Model Evaluation***
#> Input: data frame of probabilities of observed labels
#> Group column exists.
#> Observations: 102
#> Number of groups: 2
#> Observations per group: 51
#> Positive: R
#> Negative: M
#> Group: gbm
#> Positive: 24
#> Negative: 27
#> Group: ranger
#> Positive: 24
#> Negative: 27
#> ***Performance Metrics***
#> gbm Optimal Informedness = 0.722222222222222
#> ranger Optimal Informedness = 0.712962962962963
#> gbm AUC-ROC = 0.9
#> ranger AUC-ROC = 0.92
```



4. Running on Caret train output: single group, balanced data

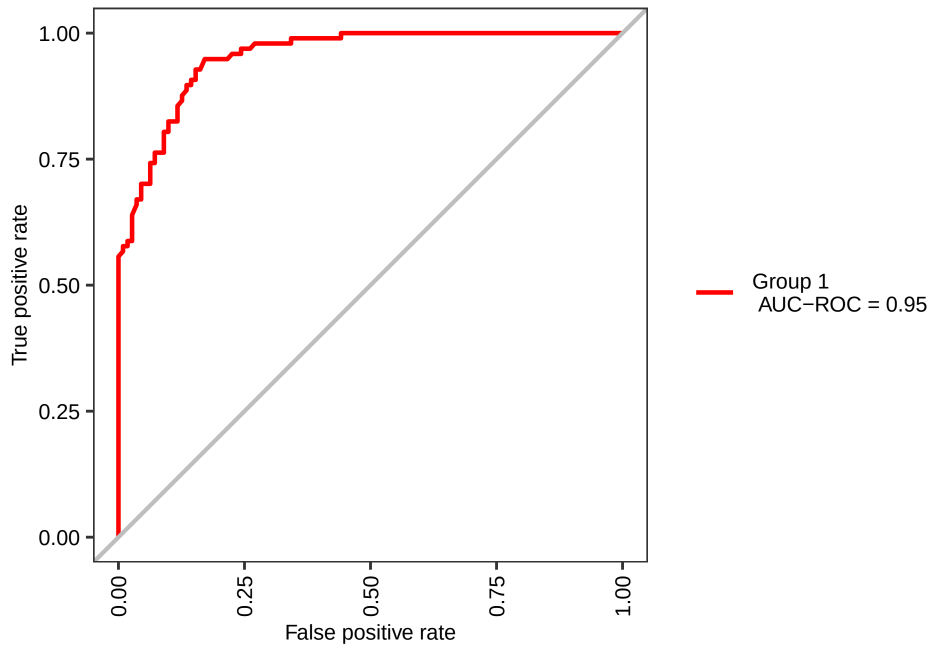
MLeval makes it very straightforward to evaluate model performance from Caret cross validation loops. We can just pass in the object produced by the Caret 'train' function.

Note, that the original Caret code is commented out from this point forward and we have saved the Caret train objects for use.

```r
library(MLeval)

## run cross validation on Sonar data
# fitControl <- trainControl(
#   method = "repeatedcv",
#   summaryFunction=twoClassSummary,
#   classProbs=T,
#   savePredictions = T)
# fit1 <- train(Class ~ ., data = Sonar,
#                 method = "ranger",
#                 trControl = fitControl,metric = "ROC",
#                 verbose = FALSE)

## evaluate
test1 <- evalm(fit1,plots='r',rlinethick=0.8,fsize=8)
#> ***MLeval: Machine Learning Model Evaluation***
#> Input: caret train function object
#> Averaging probs.
#> Group 1 type: repeatedcv
#> Observations: 208
#> Number of groups: 1
#> Observations per group: 208
#> Positive: R
#> Negative: M
#> Group: Group 1
#> Positive: 97
#> Negative: 111
#> ***Performance Metrics***
#> Group 1 Optimal Informedness = 0.777282437076251
#> Group 1 AUC-ROC = 0.95
```



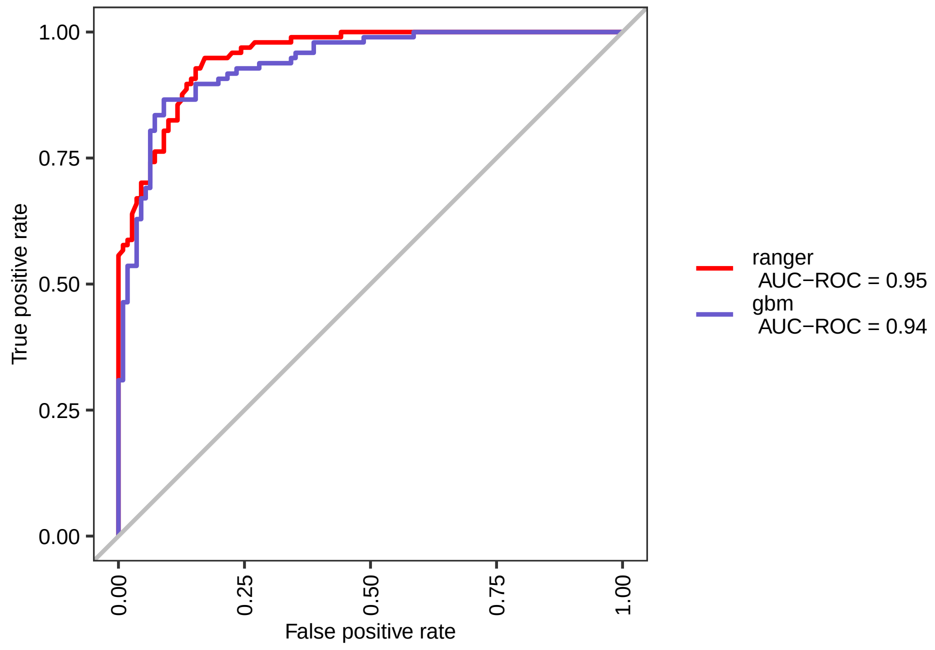
5. Running on Caret train output: multiple groups, balanced data

We can also pass a list of models run using the Caret train function into MLeval for a combined comparison.

```
library(MLeval)

# Caret train function output object -- repeated cross validation
# run caret
# fitControl <- trainControl(
#   method = "repeatedcv",
#   summaryFunction=twoClassSummary,
#   classProbs=T,
#   savePredictions = T)
# fit2 <- train(Class ~ ., data = Sonar,
#               method = "gbm",
#               trControl = fitControl,metric = "ROC",
#               verbose = FALSE)

# plot rocs
test4 <- evalm(list(fit1,fit2),gnames=c('ranger','gbm'),rlinethick=0.8,fsize=8,
              plots='r')
#> ***MLeval: Machine Learning Model Evaluation***
#> Input: caret train function object
#> Averaging probs.
#> Group 1 type: repeatedcv
#> Group 2 type: repeatedcv
#> Observations: 416
#> Number of groups: 2
#> Observations per group: 208
#> Positive: R
#> Negative: M
#> Group: ranger
#> Positive: 97
#> Negative: 111
#> Group: gbm
#> Positive: 97
#> Negative: 111
#> ***Performance Metrics***
#> ranger Optimal Informedness = 0.777282437076251
#> gbm Optimal Informedness = 0.775889291353209
#> ranger AUC-ROC = 0.95
#> gbm AUC-ROC = 0.94
```



We can see a list has been added containing the results from each model.

```
test4$optres
#> $ranger
#>                  Score        CI
#> SENS             0.948 0.88-0.98
#> SPEC             0.829 0.75-0.89
#> MCC              0.777      <NA>
#> Informedness     0.777      <NA>
#> PREC             0.829 0.75-0.89
#> NPV              0.948 0.88-0.98
#> FPR              0.171      <NA>
#> F1               0.885      <NA>
#> TP              92.000      <NA>
#> FP              19.000      <NA>
#> TN              92.000      <NA>
#> FN               5.000      <NA>
#> AUC-ROC          0.950 0.92-0.98
#> AUC-PR           0.940      <NA>
#> AUC-PRG          0.690      <NA>
#>
#> $gbm
#>                 Score        CI
#> SENS            0.866 0.78-0.92
#> SPEC            0.910 0.84-0.95
#> MCC             0.778      <NA>
#> Informedness    0.776      <NA>
#> PREC            0.894 0.82-0.94
#> NPV             0.886 0.81-0.93
#> FPR             0.090      <NA>
#> F1              0.880      <NA>
#> TP             84.000      <NA>
#> FP             10.000      <NA>
#> TN            101.000      <NA>
#> FN             13.000      <NA>
#> AUC-ROC         0.940 0.91-0.97
#> AUC-PR          0.920      <NA>
#> AUC-PRG         0.640      <NA>
```

6. Running on Caret train output: single group, imbalanced data

Very imbalanced data requires special consideration where we are interested just in the positive class. In these situations the ROC gives a overly optimistic picture of the situation because it does not directly consider false positives in the positive fraction. So we can use the precision-recall curve or the precision-recall gain curve, and the area under these as metrics. These work in the same way as a ROC, but precision vs recall is plotted while we change the probability required for a positive parameter.

Note, in the Caret example code we are running non repeated 10 fold cross validation for speed, repeated cross validation is required for reliable estimates. If not repeating, we can end up with wildly optimistic or pessimistic estimates of performance. Because the data is very imbalanced, when running Caret we must select 'prSummary' to calculate the area under precision-recall curve for each test data, this is used to select the best model from searching across all parameters.

```
library(MLeval)

# im <- twoClassSim(2000, intercept = -25, linearVars = 20)
# table(im$Class)
#
# fitControl <- trainControl(
#   method = "cv",
#   summaryFunction=prSummary,
#   classProbs=T,
#   savePredictions = T,
#   verboseIter = F)
# im_fit <- train(Class ~ ., data = im,
#                  method = "ranger",
#                  metric = "AUC",
#                  trControl = fitControl)

x <- evalm(im_fit,rlinethick=0.8,fsize=8,plots=c())
#> ***MLeval: Machine Learning Model Evaluation***
#> Input: caret train function object
#> Not averaging probs.
#> Group 1 type: cv
#> Observations: 2000
#> Number of groups: 1
#> Observations per group: 2000
#> Positive: Class2
#> Negative: Class1
#> Group: Group 1
#> Positive: 51
#> Negative: 1949
#> ***Performance Metrics***
#> Group 1 Optimal Informedness = 0.860954335556696
#> Group 1 AUC-ROC = 0.98
```

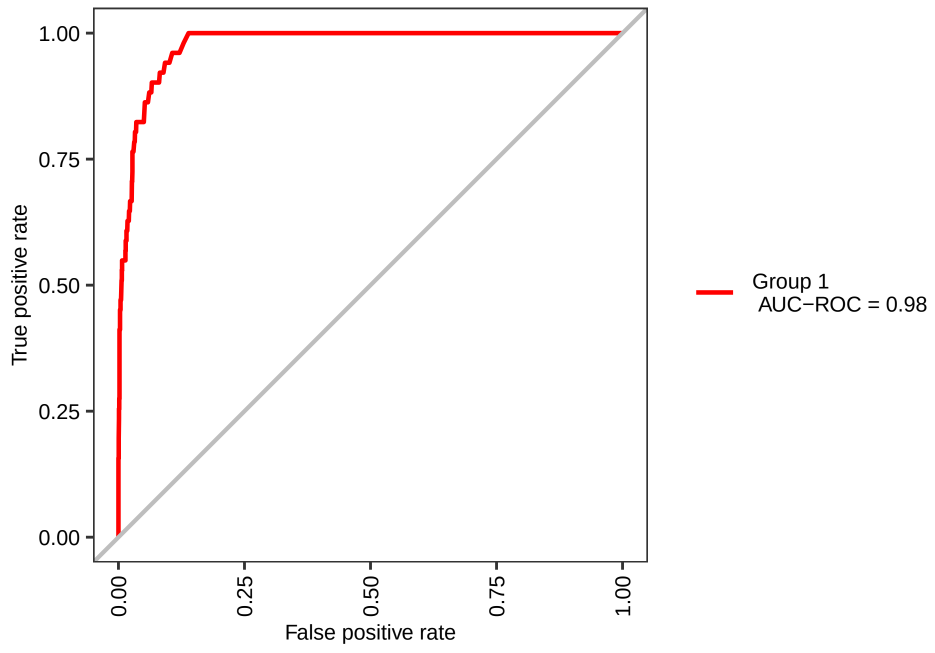
The optimised results will have been extracted for the maximal informedness operating point, informedness may not be the best metric for imbalanced data. So we could change this using the 'optimise' flag to e.g. Matthew's correlation coefficient, see ?evalm for details. The best operating point can also been extracted manually using the probabilities table output.

We can see with very imbalanced data it is possible for sensitivity and specificity to be very high, but precision to be low. The metrics that correctly pick this out, are AUC-PR, AUC-PRG, and Matthew's correlation coefficient.

```
x$optres
#> $`Group 1`
#>                 Score        CI
#> SENS            1.000    0.93-1
#> SPEC            0.861 0.84-0.88
#> MCC             0.369      <NA>
#> Informedness    0.861      <NA>
#> PREC            0.158  0.12-0.2
#> NPV             1.000       1-1
#> FPR             0.139      <NA>
#> F1              0.273      <NA>
#> TP             51.000      <NA>
#> FP            271.000      <NA>
#> TN           1678.000      <NA>
#> FN              0.000      <NA>
#> AUC-ROC         0.980 0.95-1.01
#> AUC-PR          0.610      <NA>
#> AUC-PRG         0.270      <NA>
```

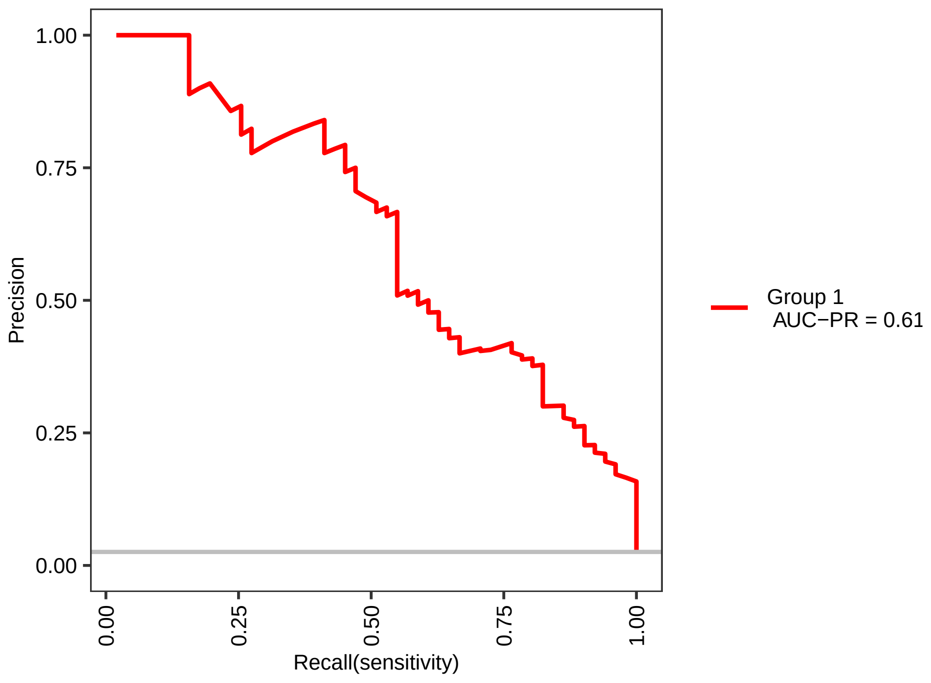
We can see the ROC here demonstrates an excellent discriminative ability, but this is based just on specificity and sensitivity. These consider how many negatives or positives have been picked up, but not how many predicted positives are in fact negatives. When negatives are much greater than positives it is possible to have a good ROC with many false positives in the predicted positive fraction. In some contexts, it may be desirable to tease this information out with precision-recall curves.

```
x$roc
```



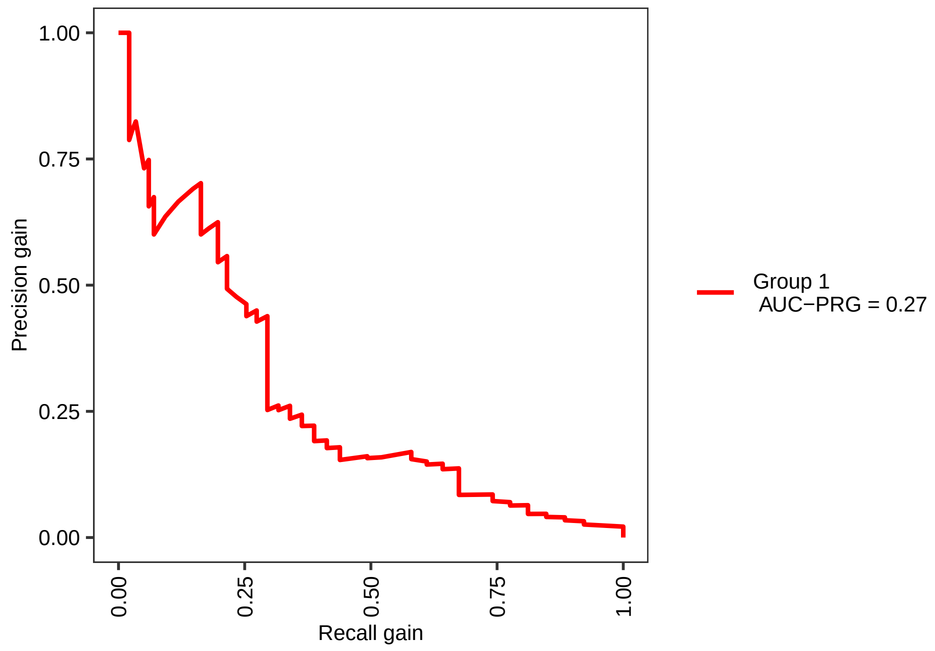
This is the precision-recall curve, the horizontal grey line corresponds to the baseline chance expectation, where this equals P/N, the positives over positives plus negatives. The baseline for precision is always the same over the sensitivity analysis, whereas for sensitivity it increases incrementally, hence the horizontal line.

```
x$proc
```



This is the precision-recall gain curve, which standardises precision against the baseline chance expectation. Flach and Meelis (2015) discuss in detail the preferable properties of precision-recall gain curves to precision-recall curves. The precision-recall gain curve is telling us here our precision is quite poor relative to the baseline. Recall, the baseline for PRG curves is zero not a diagonal line like ROC.

```
x$prg
```



7. Running on Caret train output: log-likelihood to select model

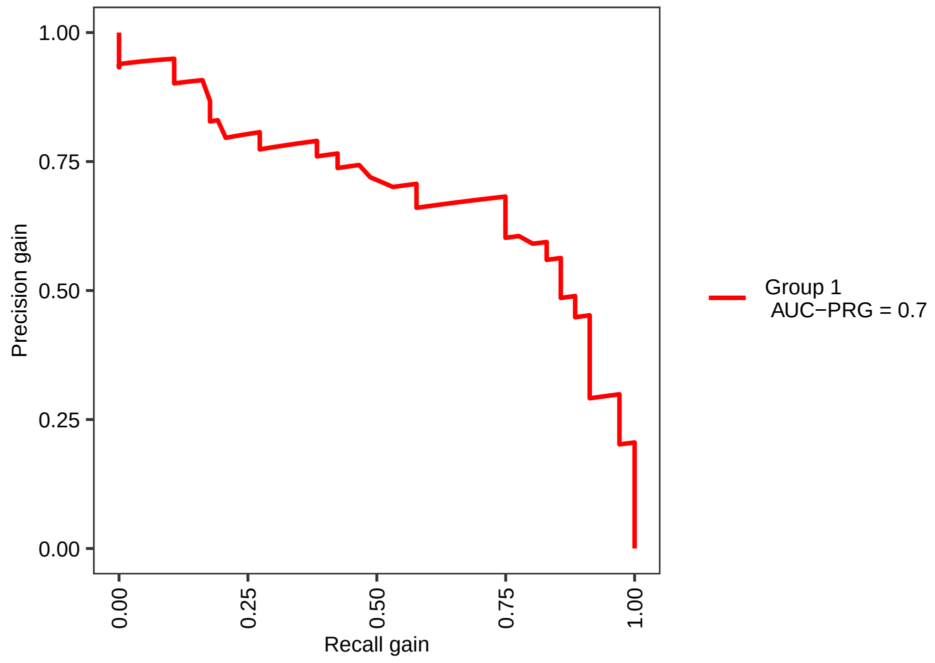
Another approach is to always use proper scores to select the best model such as the log-likelihood and the Brier score, both of which are provided by MLeval. The argument is that the ROC-AUC or PR-AUC are insensitive compared with evaluating the probabilities directly because they first convert them to TP/FP/TN/FP. After choosing the best model using log-likelihood the discriminative power of the model may be evaluated using a ROC or PR curve.

The way to do this with MLeval and Caret is as follows. Note, that with log-likelihood zero is optimal and more negative is less optimal and MLeval choosing the highest scoring model so we do not need to transform the LL function (which is from this package) output. If we were using the Brier score we would need to negative the output of this function when making the custom summary function.

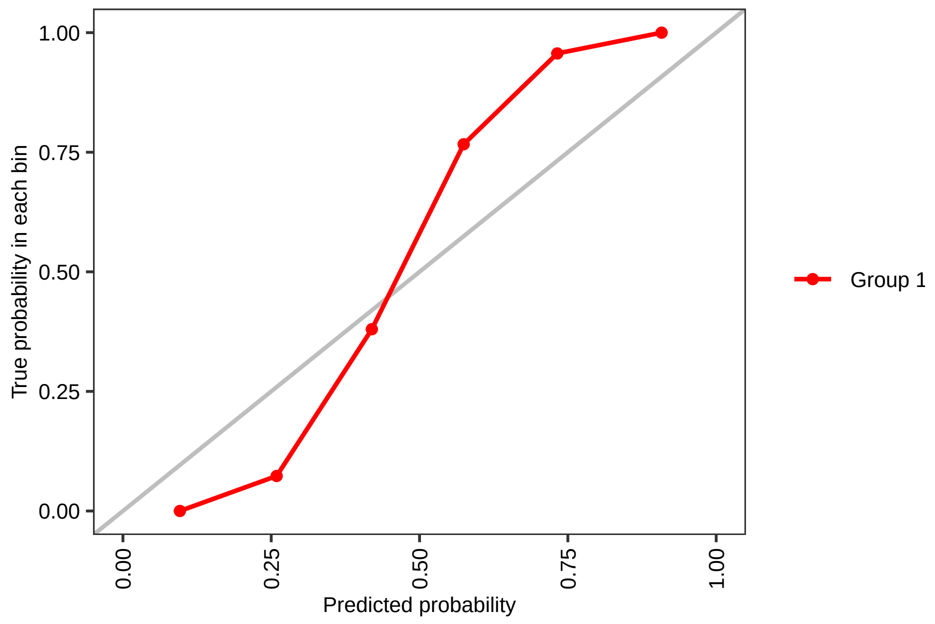
```r
library(MLeval)

# # set up custom function for the log likelihood
# LLSummary <- function(data, lev = NULL, model = NULL){
#   LLi <- LL(data,positive='R')
#   names(LLi) <- "LL"
#   out <- LLi
#   out
# }
#
# fitControl <- trainControl(
#   method = "cv",
#   summaryFunction = LLSummary,
#   classProbs=T,
#   savePredictions = T,
#   verboseIter = T)
# fit3 <- train(Class ~ ., data = Sonar,
#               method = "ranger",
#               trControl = fitControl,metric = "LL",
#               verbose = FALSE)
```

```
y <- evalm(fit3,rlinethick=0.8,fsize=8,plots=c('prg'))
#> ***MLeval: Machine Learning Model Evaluation***
#> Input: caret train function object
#> Not averaging probs.
#> Group 1 type: cv
#> Observations: 208
#> Number of groups: 1
#> Observations per group: 208
#> Positive: R
#> Negative: M
#> Group: Group 1
#> Positive: 97
#> Negative: 111
#> ***Performance Metrics***
#> Group 1 Optimal Informedness = 0.799108386737253
#> Group 1 AUC-ROC = 0.95
```



```
y$cc
```



8. Closing comments

For further information, beyond the scope of this document, we suggest to read the work of David Power who came up with informedness, he has many papers on the problems of F1 score, kappa, etc (references below). Also the 2015 paper on precision recall gain curves is a very good introduction to this kind of approach.

MLeval code is hosted in the following github (https://github.com/crj32/MLeval).

9. References

Kuhn, Max. "Building predictive models in R using the caret package." Journal of statistical software 28.5 (2008): 1-26.

Flach, Peter, and Meelis Kull. "Precision-recall-gain curves: PR analysis done right." Advances in neural information processing systems. 2015.

Powers, David Martin. "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation." (2011).

Powers, David MW. "What the F-measure doesn't measure: Features, Flaws, Fallacies and Fixes." arXiv preprint arXiv:1503.06410 (2015).