Exercise collection – Performance Evaluation

Contents

Lecture exercises

Exercise 1: performance evaluation in `mlr3`

In preparing this course you already learned about `mlr3`. If you need to refresh your knowledge you can find help at `https://mlr3book.mlr-org.com/` under 'Basics'.

a) How many performance measures do you already know? Try to explain some of them. How can you see which of them are available in `mlr3`?

b) Use the `boston_housing` regression task from `mlr3` and split the data into 50 % training data and 50 % test data while training and predicting (i. e., use the `row_ids` argument of the `train` and `predict` function). Fit a prediction model (e. g. k-NN) to the training set and make predictions for the test set.

c) Compare the performance on training and test data. Use the `score` function.

d) Now use different observations (but still 50 % of them) for the training set. How does this affect the predictions and the error estimates of the test data?

e) Use 10 fold cross-validation to estimate the performance. Hint: Use the mlr functions `rsmp` and `resample`.

Solution 1:

a) Each loss function we have learned so far to fit the model (inner loss) can also be used as performance measure (outer loss).

For classification:

- 0-1 loss (= mean misclassification error),
- Logistic loss (bernoulli loss), ...

For regression:

- L_2-loss (= mean squared error),
- L_1-loss (= mean absolute error), ...

To get a list of all measures you can use `mlr_measures`.

b)
```
# look at the task
task <- tsk("boston_housing")
task

## <TaskRegr:boston_housing> (506 x 19)
## * Target: medv
## * Properties: -
## * Features (18):
##   - dbl (13): age, b, cmedv, crim, dis, indus, lat, lon, lstat, nox,
##     ptratio, rm, zn
##   - int (3): rad, tax, tract
##   - fct (2): chas, town

n <- task$nrow

# select index vectors to subset the data randomly
set.seed(123)
train_ind <- sample(seq_len(n), 0.5*n)
test_ind <- setdiff(seq_len(n), train_ind)

# specify learner
learner <- lrn("regr.kknn", k = 3)

# train model to the training set
learner$train(task, row_ids = train_ind)

# predict on the test set
pred <- learner$predict(task, row_ids = test_ind)
pred

## <PredictionRegr> for 253 observations:
##     row_ids truth response
##           1  24.0 23.22445
##           2  21.6 19.98830
##           3  34.7 34.97419
## ---
##         504  23.9 22.22775
##         505  22.0 21.76531
##         506  11.9 20.88958
```

c)
```
# predict on the train set
pred_train <- learner$predict(task, row_ids = train_ind)
pred_train$score(list(msr("regr.mse"), msr("regr.mae")))

##  regr.mse  regr.mae
## 1.2322560 0.7564092
```

```
# predict on the test set
pred_test <- learner$predict(task, row_ids = test_ind)
pred_test$score(list(msr("regr.mse"), msr("regr.mae")))

##  regr.mse  regr.mae
## 12.424958  2.596332
```

Unsurprisingly the model performs better on the training data (smaller loss) then on the test data.

d)
```
# select different index vectors to subset the data randomly
set.seed(321)
train_ind <- sample(seq_len(n), 0.5*n)
test_ind <- setdiff(seq_len(n), train_ind)

# specify learner
learner <- lrn("regr.kknn", k = 3)

# train model to the training set
learner$train(task, row_ids = train_ind)

# predict on the test set
pred_test <- learner$predict(task, row_ids = test_ind)
pred_test

## <PredictionRegr> for 253 observations:
##     row_ids truth response
##           2  21.6 29.45474
##           5  36.2 33.14900
##           6  28.7 32.95574
## ---
##         501  16.8 19.61312
##         505  22.0 23.29286
##         506  11.9 21.28301

pred_test$score(list(msr("regr.mse"), msr("regr.mae")))

##  regr.mse  regr.mae
## 12.507468  2.458798
```

Effect: We will predict different observations since the test set is different. The same observations get a slightly different prediction (e.g. observation with id 2). This affects the final error estimation.

e)
```
rdesc <- rsmp("cv", folds = 10)
r <- resample(task, learner, rdesc)

## INFO  [10:39:01.030] [mlr3]  Applying learner 'regr.kknn' on task 'boston_housing' (iter 9/10)
## INFO  [10:39:01.080] [mlr3]  Applying learner 'regr.kknn' on task 'boston_housing' (iter 10/10)
## INFO  [10:39:01.105] [mlr3]  Applying learner 'regr.kknn' on task 'boston_housing' (iter 7/10)
## INFO  [10:39:01.125] [mlr3]  Applying learner 'regr.kknn' on task 'boston_housing' (iter 6/10)
## INFO  [10:39:01.144] [mlr3]  Applying learner 'regr.kknn' on task 'boston_housing' (iter 4/10)
## INFO  [10:39:01.164] [mlr3]  Applying learner 'regr.kknn' on task 'boston_housing' (iter 8/10)
## INFO  [10:39:01.184] [mlr3]  Applying learner 'regr.kknn' on task 'boston_housing' (iter 1/10)
## INFO  [10:39:01.208] [mlr3]  Applying learner 'regr.kknn' on task 'boston_housing' (iter 5/10)
## INFO  [10:39:01.227] [mlr3]  Applying learner 'regr.kknn' on task 'boston_housing' (iter 3/10)
## INFO  [10:39:01.246] [mlr3]  Applying learner 'regr.kknn' on task 'boston_housing' (iter 2/10)

r$aggregate(list(msr("regr.mse"), msr("regr.mae")))
```

```
##  regr.mse  regr.mae
## 10.045363  2.229458
```

Exercise 2: train vs test error

The Satellite dataset consists of pixels in 3x3 neighbourhoods in a satellite image, where each pixel is described by 4 spectral values, and the classification label of the central pixel. (for further information see `?Satellite`) We fit a k-NN model to predict the class of the middle pixel. The performance is evaluated with the mmce.
Look at the following R code and output: The performance is estimated in different ways: using training data, test data and then with cross validation. How do the estimates differ and why? Which one should be used?

```
library(mlr3)
library(mlr3learners)
library(mlbench)

data(Satellite)
satellite_task <-
  TaskClassif$new(id = "satellite_task",
                  backend = Satellite,
                  target = "classes")
knn_learner <- lrn("classif.kknn", k = 3)

# Train and test subsets:
set.seed(42)
train_indices <-
  sample.int(nrow(Satellite), size = 0.8 * nrow(Satellite))
test_indices <- setdiff(1:nrow(Satellite), train_indices)

# Training data performance estimate
knn_learner$train(task = satellite_task, row_ids = train_indices)
pred <-
  knn_learner$predict(task = satellite_task, row_ids = train_indices)
pred$score()

## classif.ce
##          0

# Test data performance estimate
pred <-
  knn_learner$predict(task = satellite_task, row_ids = test_indices)
pred$score()

## classif.ce
## 0.09246309

# CV performance estimate
rdesc <- rsmp("cv", folds = 10)
res <- resample(satellite_task, knn_learner, rdesc)

## INFO  [10:39:02.376] [mlr3]  Applying learner 'classif.kknn' on task 'satellite_task' (iter 1/10)
## INFO  [10:39:02.508] [mlr3]  Applying learner 'classif.kknn' on task 'satellite_task' (iter 4/10)
## INFO  [10:39:02.638] [mlr3]  Applying learner 'classif.kknn' on task 'satellite_task' (iter 7/10)
## INFO  [10:39:02.871] [mlr3]  Applying learner 'classif.kknn' on task 'satellite_task' (iter 5/10)
## INFO  [10:39:02.995] [mlr3]  Applying learner 'classif.kknn' on task 'satellite_task' (iter 3/10)
```

```
## INFO  [10:39:03.118] [mlr3]  Applying learner 'classif.kknn' on task 'satellite_task' (iter 9/10)
## INFO  [10:39:03.241] [mlr3]  Applying learner 'classif.kknn' on task 'satellite_task' (iter 6/10)
## INFO  [10:39:03.366] [mlr3]  Applying learner 'classif.kknn' on task 'satellite_task' (iter 10/10)
## INFO  [10:39:03.491] [mlr3]  Applying learner 'classif.kknn' on task 'satellite_task' (iter 2/10)
## INFO  [10:39:03.618] [mlr3]  Applying learner 'classif.kknn' on task 'satellite_task' (iter 8/10)
```

```
res$score()
```

```
##                  task        task_id                   learner   learner_id
##  1: <TaskClassif[46]> satellite_task <LearnerClassifKKNN[32]> classif.kknn
##  2: <TaskClassif[46]> satellite_task <LearnerClassifKKNN[32]> classif.kknn
##  3: <TaskClassif[46]> satellite_task <LearnerClassifKKNN[32]> classif.kknn
##  4: <TaskClassif[46]> satellite_task <LearnerClassifKKNN[32]> classif.kknn
##  5: <TaskClassif[46]> satellite_task <LearnerClassifKKNN[32]> classif.kknn
##  6: <TaskClassif[46]> satellite_task <LearnerClassifKKNN[32]> classif.kknn
##  7: <TaskClassif[46]> satellite_task <LearnerClassifKKNN[32]> classif.kknn
##  8: <TaskClassif[46]> satellite_task <LearnerClassifKKNN[32]> classif.kknn
##  9: <TaskClassif[46]> satellite_task <LearnerClassifKKNN[32]> classif.kknn
## 10: <TaskClassif[46]> satellite_task <LearnerClassifKKNN[32]> classif.kknn
##             resampling resampling_id iteration              prediction
##  1: <ResamplingCV[19]>            cv         1 <PredictionClassif[19]>
##  2: <ResamplingCV[19]>            cv         2 <PredictionClassif[19]>
##  3: <ResamplingCV[19]>            cv         3 <PredictionClassif[19]>
##  4: <ResamplingCV[19]>            cv         4 <PredictionClassif[19]>
##  5: <ResamplingCV[19]>            cv         5 <PredictionClassif[19]>
##  6: <ResamplingCV[19]>            cv         6 <PredictionClassif[19]>
##  7: <ResamplingCV[19]>            cv         7 <PredictionClassif[19]>
##  8: <ResamplingCV[19]>            cv         8 <PredictionClassif[19]>
##  9: <ResamplingCV[19]>            cv         9 <PredictionClassif[19]>
## 10: <ResamplingCV[19]>            cv        10 <PredictionClassif[19]>
##     classif.ce
##  1: 0.09627329
##  2: 0.07919255
##  3: 0.08540373
##  4: 0.09472050
##  5: 0.10559006
##  6: 0.08553655
##  7: 0.08864697
##  8: 0.10108865
##  9: 0.08864697
## 10: 0.10419907
```

```
res$aggregate()
```

```
## classif.ce
## 0.09292983
```

Solution 2:

- The training performance is too optimistic (mmce of 0), because the mmce is higher on new data.
- The test performance is unbiased (if the final model is only trained on the training data), but it depends on the split, as can be seen in the CV folds: Each CV fold represents a training test split and the mmce measure varies between folds.

- The CV estimate averages over the different splits and gives an slightly pessimistic, more robust estimate.
- The CV estimate is preferable over the other two, but more computationally expensive.

Exercise 3: generalization error

Shortly answer the following questions:

(a) What is the difference between inner and outer loss?

(b) Which model is more likely to overfit the training data:

- k-NN with 1 or with 10 neighbours?
- Logistic regression with 10 or 20 features?
- LDA or QDA?

(c) Which of the following methods yield an unbiased generalization error estimate? Performance estimation ...

- on training data
- on test data
- on training and test data combined
- using cross validation
- using subsampling

(d) Which problem does resampling of training and test data solve?

(e) Which problem does nested resampling solve?

Solution 3:

- The inner loss is the loss that is optimized directly by the machine learning model. The outer loss is the loss (or performance measurement) used to evaluate the model.
- Which model is more likely to overfit the training data:
  - k-NN with 1 or with 10 neighbors? **1 neighbor**, because it's an exact memorization of training data.
  - Logistic regression with 10 or 20 features? **20 features**, because the more features, the more coefficients the learner estimates. More coefficients mean more degrees of freedom, which make overfitting more likely.
  - LDA or QDA? **QDA**, because it has more parameters to possibly overfit the data. LDA is more likely to underfit more complex relationships.
- Which of the following methods yield an unbiased generalization error estimate? Performance estimation ...
  - on training data: **Biased, too optimistic**
  - on test data: **Unbiased / Biased, too pessimistic** (Test data is *not included* / *included* in the final model)
  - on training and test data combined: **Biased, too optimistic** (But a little bit less than only using training data).

  - using cross validation: **Biased, too pessimistic** (The higher the ratio of folds / number of observation, the smaller the pessimistic bias)
  - using subsampling: **Biased, too pessimistic** (The smaller the subsampling rate, the larger the pessimistic bias)

- Resampling strategies solve the problem that comes from the randomness of the training and test data split: Error estimation using a single split has a high variance. Resampling estimates are more robust because they average over different splits.
- Nested resampling solves the problem of simultaneously conducting tuning/model selection and performance estimation. When we use the performance estimates from the same data that were used for model selection (as done in simple, not-nested resampling), the final error estimate is too optimistic.

Exercise 4: ROC precision

Derive the formula of precision in terms of sensitivity and specificity and prevalence. Hint: prevalence is the percentage of the positive class.

Solution 4:

Derive the formula of precision in terms of sensitivity and specificity and prevalence. Hint: prevalence is the percentage of the positive class.

Hint: Use bayes formula to revert the conditional probability, you could also use other methods.

precision = P(T[+])/Pr[+]), conditional on that an instance is classified to have cancer, the probability that it really has cancer. Pr[+] means predicted positive, T[+] means the true lable, and similarly for negative.

$$\begin{aligned} P(T(+)/Pr(+)) &= P(T[+]Pr[+])/P(Pr[+]) \\ &= P(T[+]Pr[+])/(P(Pr[+]T[+]) + P(Pr[+]T[-])) \\ &= P(Pr[+]/T[+])P(T+)/(P(Pr[+]/T[+])P(T+) + P(Pr[+]/T[-])P(T-)) \\ &= Sensitivity \times Prevalence/(Sensitivity \times Prevalence + (1 - Specificity)(1 - Prevelance)) \end{aligned} \tag{1}$$

Exercise 5: confusion matrix, ROC & AUC

Given are the results of a scoring algorithm and the associated *true* classes of 10 observations:

ID	Actual Class	Score
1	0	0.33
2	0	0.27
3	1	0.11
4	1	0.38
5	1	0.17
6	0	0.63
7	1	0.62
8	1	0.33
9	0	0.15
10	0	0.57

a) Create a confusion matrix assuming the decision boundary at 0.5.

b) Calculate: precision, sensitivity, negative predictive value, specificity, accuracy, error rate and F-measure.

c) Draw the ROC curve and interpret it. Feel free to use R for the drawing.

d) Calculate the AUC.

Solution 5:

a) First, sort the table:

ID	Actual Class	Score	Predicted Class
6	0	0.63	1
7	1	0.62	1
10	0	0.57	1
4	1	0.38	0
1	0	0.33	0
8	1	0.33	0
2	0	0.27	0
5	1	0.17	0
9	0	0.15	0
3	1	0.11	0

	Actual Class - 0	Actual Class - 1
Prediction - 0	3	4
Prediction - 1	2	1

so we get

FN	FP	TN	TP
4	2	3	1

b)

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{1}{3}$$

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{1}{5}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \frac{4}{10}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} = \frac{3}{5}$$

$$\text{Error Rate} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \frac{6}{10}$$

$$\text{F-measure} = \frac{2 \cdot \text{Precision} \cdot \text{Sensitivity}}{\text{Precision} + \text{Sensitivity}} = 0.25$$

$$\text{Negative Predictive Value} = \frac{\text{TN}}{\text{TN} + \text{FN}} = \frac{3}{7}$$

c) First we sort the results by the score:

	true_labels	scores
6	0	0.63
7	1	0.62
10	0	0.57
4	1	0.38
1	0	0.33
8	1	0.33
2	0	0.27
5	1	0.17
9	0	0.15
3	1	0.10

Here we see that $\frac{1}{n_+} = \frac{1}{5} = 0.2$ and $\frac{1}{n_-} = \frac{1}{5} = 0.2$. Now we follow the algorithm as described in the lecture slides:

(i) Set $\alpha = 1$, so we start in $(0, 0)$; we predict everything as 1.

(ii) Set threshold $\tau = 0.625$ yields TPR 0 and FPR $0 + \frac{1}{n_-} = 0.2$. (Obs. 6 is "0")

(iii) Set threshold $\tau = 0.6$ yields TPR $0 + \frac{1}{n_+} = 0.2$ and FPR 0.2. (Obs. 7 is "1")

(iv) Set threshold $\tau = 0.5$ yields TPR 0.2 and FPR $0.2 + \frac{1}{n_-} = 0.4$. (Obs. 10 is "0")

(v) Set threshold $\tau = 0.35$ yields TPR $0.2 + \frac{1}{n_+} = 0.4$ and FPR 0.4. (Obs. 4 is "1")

(vi) Set threshold $\tau = 0.3$ yields TPR $0.4 + \frac{1}{n_+} = 0.6$ and FPR $0.4 + \frac{1}{n_-} = 0.6$. (Obs. 1/8 is "0"/"1")

(vii) Set threshold $\tau = 0.2$ yields TPR 0.6 and FPR $0.6 + \frac{1}{n_-} = 0.8$. (Obs. 2 is "0")

(viii) Set threshold $\tau = 0.16$ yields TPR $0.6 + \frac{1}{n_+} = 0.8$ and FPR 0.8. (Obs. 5 is "1")

(ix) Set threshold $\tau = 0.14$ yields TPR 0.8 and FPR $0.8 + \frac{1}{n_-} = 1$. (Obs. 9 is "0")

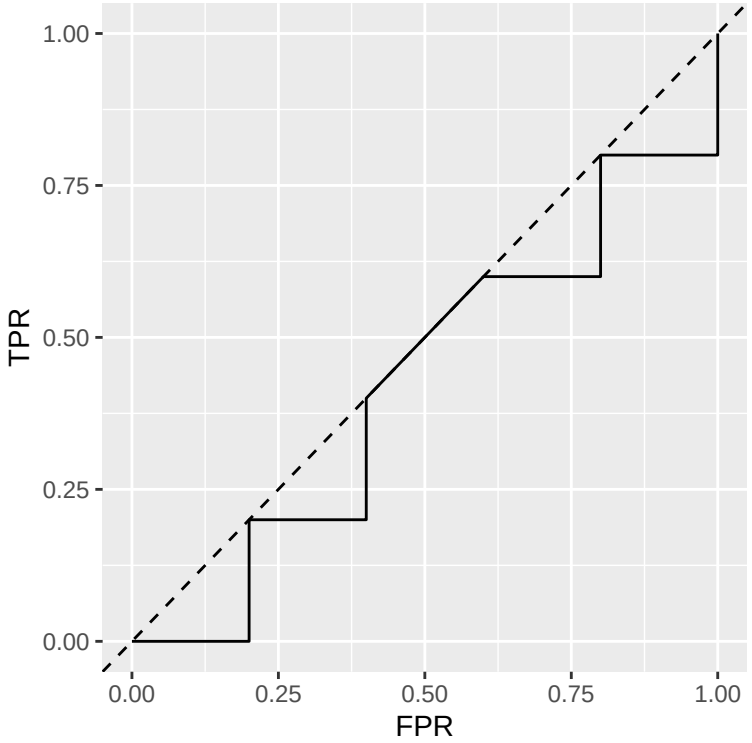
(x) Set threshold $\tau = 0.09$ yields TPR $0.8 + \frac{1}{n_+} = 1$ and FPR 1. (Obs. 3 is "1")

Therefore we get the polygonal path consisting of the ordered list of vertices

$$(0,0), (0.2,0), (0.2,0.2), (0.4,0.2), (0.4,0.4), (0.6,0.6), (0.8,0.6), (0.8,0.8), (1,0.8), (1,1).$$

```
library(ggplot2)
roc_data <- data.frame(TPR = c(0, 0,   0.2, 0.2, 0.4, 0.6, 0.6, 0.8, 0.8, 1),
                       FPR = c(0, 0.2, 0.2, 0.4, 0.4, 0.6, 0.8, 0.8,  1,  1))

ggplot(roc_data, aes(x = FPR, y = TPR)) + geom_line() +
  geom_abline(slope = 1, intercept = 0, linetype = 'dashed')
```



We see that the resulting ROC lies below the line from the origin with a slope of 1, which represents a random classifier, i.e., the scoring algorithm performs worse than a random classifier. If this happens while evaluating the training data, the labels of the scoring algorithm should be inverted.

d) We can compute the AUC (*area under the curve*) by looking at the ROC, s.t.

$$AUC = 0.5 - 4 \cdot (0.2 \cdot 0.2 \cdot 0.5) = 0.42.$$

Questions from past exams

Exercise 6: WS2020/21, main exam, question 3

Consider a binary classification algorithm that yielded the following results on 8 observations. The table shows true classes and predicted probabilities for class 1:

ID	True Class	Prediction
1	1	0.50
2	0	0.01
3	0	0.90
4	0	0.55
5	1	0.10
6	1	0.72
7	1	0.70
8	1	0.99

(a) Draw the ROC curve of the classifier manually. Explain every step thoroughly, and make sure to annotate the axes of your plot.

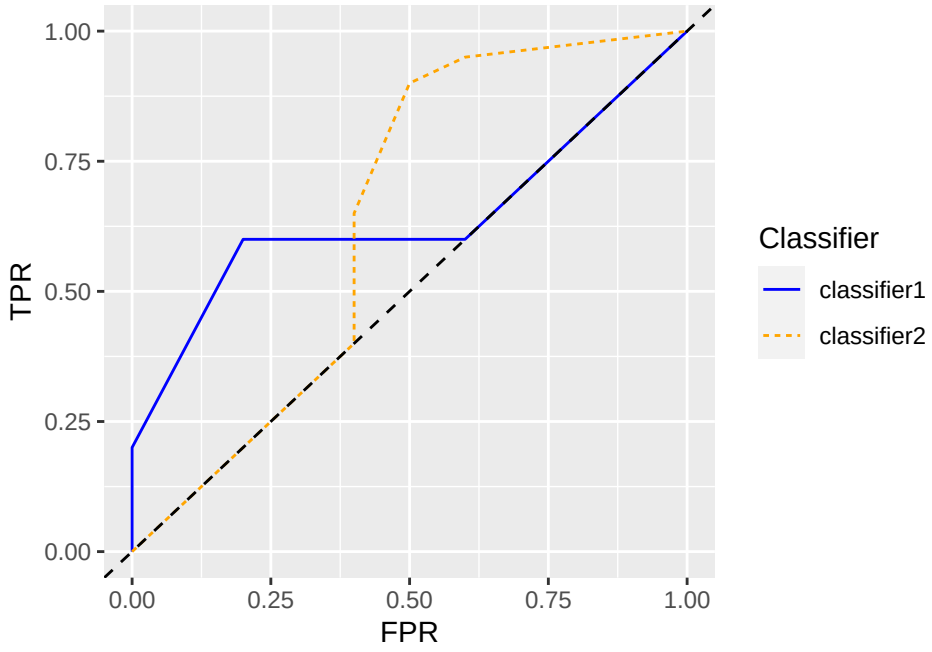
(b) Calculate the AUC of the classifier. Explain every step of your computation thoroughly.

(c) Calculate the partial AUC for FPR $\leq 1/3$. Explain every step of your computation thoroughly.

(d) Create a confusion matrix assuming a threshold of 0.75. Point out which values correspond to true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

(e) Compute sensitivity, specificity, negative predictive value, positive predictive value, accuracy and F1-score. State the respective formulas first.

(f) In the following plot, you see the ROC curves of two different classifiers with similar AUC's. Describe a practical situation where you would prefer classifier 1 over classifier 2 and explain why. (Note: The data in this question is not related to the data of the above questions.)



Solution 6:

(a) Scores:

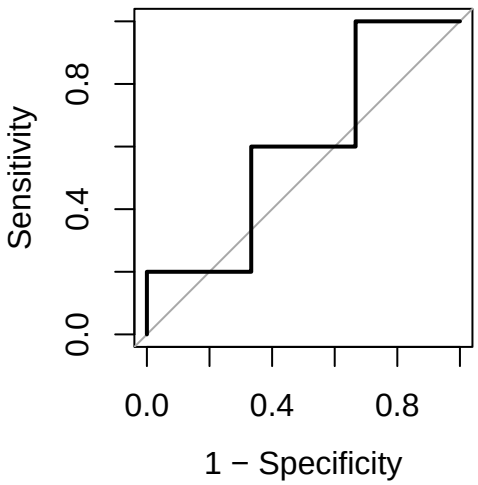
	true_labels	scores
8	1	0.99
3	0	0.90
6	1	0.72
7	1	0.70
4	0	0.55
1	1	0.50
5	1	0.10
2	0	0.01

Here we see that $\frac{1}{n_+} = \frac{1}{5} = 0.2$ and $\frac{1}{n_-} = \frac{1}{3}$. Now we follow the algorithm as described in the lecture slides:

- Set $\tau = 1$, so we start in $(0, 0)$; we predict everything as 1.
- Set $\tau = 0.95$ yields TPR $0 + \frac{1}{n_+} = 0.2$ and FPR 0. (Obs. 8 is '1')
- Set $\tau = 0.8$ yields TPR 0.2 and FPR $0 + \frac{1}{n_-} = 1/3$. (Obs. 3 is '0')
- Set $\tau = 0.71$ yields TPR $0.2 + \frac{1}{n_+} = 0.4$ and FPR 1/3. (Obs. 6 is '1')
- Set $\tau = 0.6$ yields TPR $0.4 + \frac{1}{n_+} = 0.6$ and FPR 1/3. (Obs. 7 is '1')
- Set $\tau = 0.52$ yields TPR 0.6 and FPR $1/3 + \frac{1}{n_-} = 2/3$. (Obs. 4 is '0')
- Set $\tau = 0.3$ yields TPR $0.6 + \frac{1}{n_+} = 0.8$ and FPR 2/3. (Obs. 1 is '1')
- Set $\tau = 0.05$ yields TPR $0.8 + \frac{1}{n_+} = 1$ and FPR 2/3. (Obs. 5 is '1')
- Set $\tau = 0$ yields TPR 1 and FPR $2/3 + \frac{1}{n_-} = 1$. (Obs. 2 is '0')

Therefore we get the polygonal path consisting of the ordered list of vertices

$$(0, 0), (0, 0.2), (1/3, 0.2), (1/3, 0.4), (1/3, 0.6), (2/3, 0.6), (2/3, 0.8), (2/3, 1), (1, 1).$$



(b) The AUC is the sum of three rectangles: $0.2 \cdot 1/3 + 0.6 \cdot 1/3 + 1 \cdot 1/3 = 0.6$

(c) The partial AUC is the area under the curve that is left from FPR = 1/3, so it is just the first rectangle: $0.2 \cdot 1/3 = 1/15$

(d)

	Actual Class - 0	Actual Class - 1
Prediction - 0	2	4
Prediction - 1	1	1

so we get

FN	FP	TN	TP
4	1	2	1

(e)

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{1}{5}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} = \frac{2}{3}$$

$$\text{Negative Predictive Value} = \frac{\text{TN}}{\text{TN} + \text{FN}} = \frac{1}{3}$$

$$\text{Positive Predictive Value} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{1}{2}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \frac{3}{8}$$

$$\text{F1-score} = \frac{2 \cdot \text{PPV} \cdot \text{Sensitivity}}{\text{PPV} + \text{Sensitivity}} = (0.2)/(0.7) = 2/7$$

(f) Important point is to understand that classifier 1 does better in the 'high scores'. Or: For some threshold below 0.5 the precision is far better for classifier 1 than for classifier 2. For example if you want to output a certain number of 'most promising customers' this could be a good idea.

Exercise 7: WS2020/21, retry exam, question 3

Consider a binary classification algorithm that yielded the following results on 8 observations. The table shows true classes and predicted probabilities for class 1:

ID	True Class	Prediction
1	1	0.30
2	0	0.91
3	0	0.03
4	0	0.55
5	0	0.45
6	0	0.65
7	1	0.71
8	1	0.98

(a) Draw the ROC curve of the classifier manually. Compute all relevant numbers explicitly, state the respective formulas first, and make sure to annotate the axes of your plot.

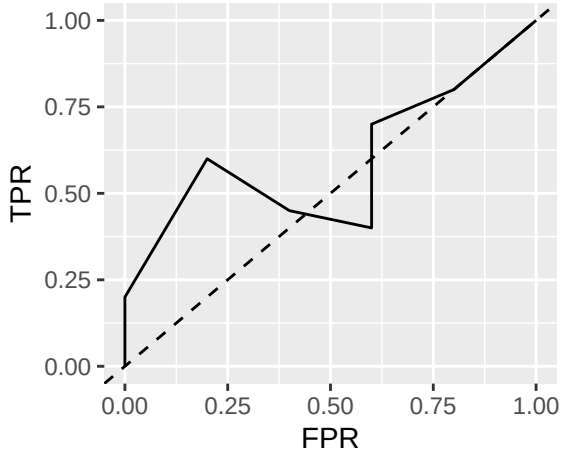
(b) Calculate the AUC of the classifier. Explain every step of your computation thoroughly.

(c) Calculate the partial AUC for TPR $\geq 2/3$. Explain every step of your computation thoroughly.

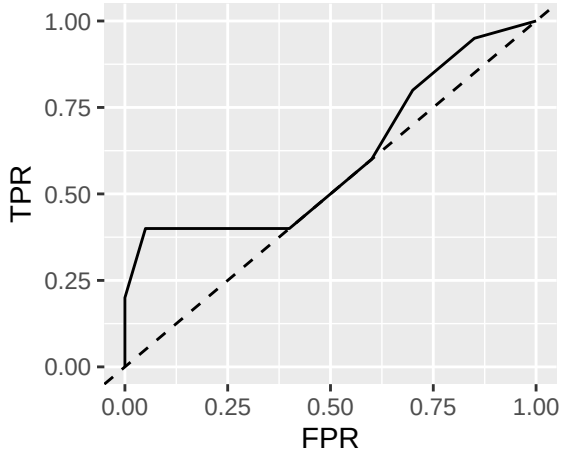
(d) Create a confusion matrix assuming a threshold of 0.7. Point out which values correspond to true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

(e) Compute sensitivity, specificity, negative predictive value, positive predictive value, accuracy and F1-score. State the respective formulas first.

(f) Now look at the following plot. Explain thoroughly why the solid line cannot be the ROC curve of a classifier.



(g) Imagine you are working in the marketing department of a large company. Your colleagues are developing a marketing campaign where they will call selected customers by phone in order to advertise a new product. Your company has 1,000,000 customers in the database and the budget of the marketing campaign allows to call 1,000 of these customers. Now it is your job to select the most promising 1,000 customers, i.e., those who will most likely buy the new product after having received the advertising phone call. Luckily, you have a large amount of similar data from older marketing campaigns that are representative for this campaign and can be used to train a supervised classification model with the target variable indicating if the customer did buy the product (1) or not (0). You train a random forest on the 1,000,000 observations and get the following cross-validated ROC curve. Assuming a balanced target variable: Do you think this model is fairly good for your purpose? Explain why and describe, how you would proceed from here to provide your colleagues with the 1,000 most promising customers.



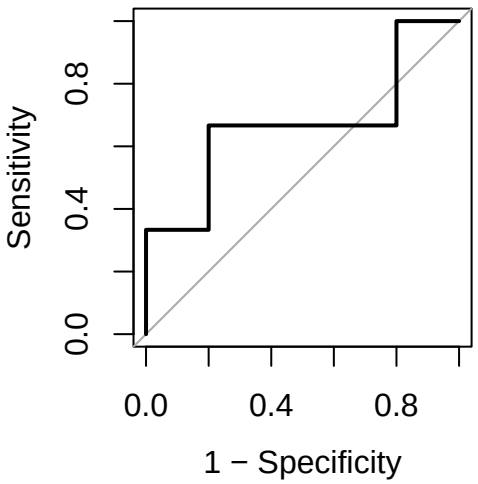
Solution 7:

(a) First we sort the results by score:

	true_labels	scores
8	1	0.98
2	0	0.91
7	1	0.71
6	0	0.65
4	0	0.55
5	0	0.45
1	1	0.30
3	0	0.03

Here we see that $\frac{1}{n_-} = \frac{1}{5} = 0.2$ and $\frac{1}{n_+} = \frac{1}{3}$. Now we follow the algorithm as described in the lecture slides:

- Set $\tau = 1$, so we start in $(0, 0)$; we predict everything as 1.
- Set $\tau = 0.95$ yields TPR $0 + \frac{1}{n_+} = 1/3$ and FPR 0. (Obs. 8 is '1')
- Set $\tau = 0.9$ yields TPR $1/3$ and FPR $0 + \frac{1}{n_-} = 0.2$. (Obs. 2 is '0')
- Set $\tau = 0.70$ yields TPR $1/3 + \frac{1}{n_+} = 2/3$ and FPR 0.2. (Obs. 7 is '1')
- Set $\tau = 0.6$ yields TPR $2/3$ and FPR $0.2 + \frac{1}{n_-} = 0.4$. (Obs. 6 is '0')
- Set $\tau = 0.5$ yi´elds TPR $2/3$ and FPR $0.4 + \frac{1}{n_-} = 0.6$. (Obs. 4 is '0')
- Set $\tau = 0.4$ yields TPR $2/3$ and FPR $0.6 + \frac{1}{n_-} = 0.8$. (Obs. 5 is '0')
- Set $\tau = 0.1$ yields TPR $2/3 + \frac{1}{n_+} = 1$ and FPR 0.8. (Obs. 1 is '1')
- Set $\tau = 0$ yields TPR 1 and FPR $0.8 + \frac{1}{n_-} = 1$. (Obs. 3 is '0')



(b) The AUC is the sum of three rectangles: $0.2 \cdot 1/3 + 0.6 \cdot 2/3 + 1 \cdot 0.2 = 2/3$

(c) The partial AUC is the area under the curve that is above TPR $= 2/3$, so it is just the small rectangle in the upper right corner: $0.2 \cdot 1/3 = 1/15$

(d)

	Actual Class - 0	Actual Class - 1
Prediction - 0	4	1
Prediction - 1	1	2

so we get

FN	FP	TN	TP
1	1	4	2

(e)

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{2}{3}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} = \frac{4}{5}$$

$$\text{Negative Predictive Value} = \frac{\text{TN}}{\text{TN} + \text{FN}} = \frac{4}{5}$$

$$\text{Positive Predictive Value} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{2}{3}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \frac{6}{8}$$

$$\text{F1-score} = \frac{2 \cdot \text{PPV} \cdot \text{Sensitivity}}{\text{PPV} + \text{Sensitivity}} = (8/9)/(4/3) = 2/3$$

(f) TPR and FPR are both metrics that increase monotonically as the threshold c traverses from 1 to 0. The plot shows two instances of diminishing TPR, which would mean that, at the corresponding threshold, an observation that had previously been correctly classified as positive was not detected anymore. This is not possible with a binarization threshold.

(g) Yes. I would order the customers wrt the scores, then roughly the first 200,000 customers would be true positives (since the ROC is based on 1,000,000 customers and true class is balanced) and I would just select the top 1,000 customers. It doesn't matter that the classifiers gets bad later.

Ideas & exercises from other sources