Solution 1: Performance Measures

(a) Given the following confusion matrices

$$M_1 = \begin{pmatrix} 0 & 10 \\ 0 & 990 \end{pmatrix}, \quad M_2 = \begin{pmatrix} 10 & 0 \\ 10 & 980 \end{pmatrix}, \quad M_3 = \begin{pmatrix} 10 & 10 \\ 0 & 980 \end{pmatrix},$$

each of which corresponds to a classifier. Compute the accuracy, F_1 score, G measure/mean, BAC and MCC of each classifier.

Accuracy:

Accuracy is computed by $\rho_{ACC} = \frac{TP+TN}{TP+TN+FN+FP}$. Thus, we get

$$\rho_{ACC}(M_1) = \frac{990}{1000} = 0.99, \quad \rho_{ACC}(M_2) = \frac{990}{1000} = 0.99, \quad \rho_{ACC}(M_3) = \frac{990}{1000} = 0.99.$$

F_1 score:

F_1 score is computed by $\rho_{F_1} = 2 \cdot \frac{\rho_{PPV} \cdot \rho_{TPR}}{\rho_{PPV} + \rho_{TPR}}$, where $\rho_{PPV} = \frac{TP}{TP+FP}$ and $\rho_{TPR} = \frac{TP}{TP+FN}$. Thus, we get for the precision (PPV)

$$\rho_{PPV}(M_1) = \frac{0}{10} = 0, \quad \rho_{PPV}(M_2) = \frac{10}{10} = 1, \quad \rho_{PPV}(M_3) = \frac{10}{20} = 1/2,$$

and the recall (TPR)

$$\rho_{TPR}(M_1) = \frac{0}{0} = 0, \quad \rho_{TPR}(M_2) = \frac{10}{20} = 1/2, \quad \rho_{TPR}(M_3) = \frac{10}{10} = 1,$$

so that

$$\rho_{F_1}(M_1) = 0, \quad \rho_{F_1}(M_2) = 2 \cdot \frac{1 \cdot 0.5}{1 + 0.5} = 2/3, \quad \rho_{F_1}(M_3) = 2 \cdot \frac{1 \cdot 0.5}{1 + 0.5} = 2/3.$$

Note: Here, we used the convention that $0/0 = 0$.

G score:

G score is computed by $\rho_G = \sqrt{\rho_{PPV} \cdot \rho_{TPR}}$. Thus, with the above

$$\rho_G(M_1) = 0, \quad \rho_G(M_2) = \sqrt{1 \cdot 0.5} = \sqrt{0.5} \approx 0.7071, \quad \rho_G(M_3) = \sqrt{1 \cdot 0.5} = \sqrt{0.5} \approx 0.7071.$$

G mean:

G mean is computed by $\rho_{G_m} = \sqrt{\rho_{TNR} \cdot \rho_{TPR}}$, where $\rho_{TNR} = \frac{TN}{TN+FP}$ and $\rho_{TPR} = \frac{TP}{TP+FN}$. For the TNR we have

$$\rho_{TNR}(M_1) = \frac{990}{1000} = 0.99, \quad \rho_{TNR}(M_2) = \frac{980}{980} = 1, \quad \rho_{TNR}(M_3) = \frac{980}{990} \approx 0.9899,$$

Thus, with the above

$$\rho_{G_m}(M_1) = 0, \quad \rho_{G_m}(M_2) = \sqrt{1 \cdot 0.5} = \sqrt{0.5} \approx 0.7071, \quad \rho_{G_m}(M_3) = \sqrt{1 \cdot \frac{980}{990}} \approx 0.9949.$$

BAC:

BAC is computed by $\rho_{BAC} = \frac{\rho_{TNR} + \rho_{TPR}}{2}$. Thus, with the above

$$\rho_{BAC}(M_1) = \frac{0 + 990/1000}{2} \approx 0.5, \quad \rho_{BAC}(M_2) = \frac{1 + 0.5}{2} = 0.75, \quad \rho_{BAC}(M_3) = \frac{1 + \frac{980}{990}}{2} \approx 0.9949.$$

MCC: MCC is computed by $\rho_{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FN)(TP+FP)(TN+FN)(TN+FP)}}$. Thus, with the above

$$\rho_{MCC}(M_1) = 0, \quad \rho_{MCC}(M_2) = \frac{10 \cdot 980 - 0}{\sqrt{20 \cdot 10 \cdot 990 \cdot 980}} = 0.7035265, \quad \rho_{MCC}(M_3) = \frac{10 \cdot 980 - 0}{\sqrt{20 \cdot 10 \cdot 990 \cdot 980}} = 0.7035265.$$

(b) What are the population counterparts of the class-specific variants in the multiclass setting of true positive rate, positive predictive value and true negative rate?

True positive rate:

In the multiclass classification setting ($\mathcal{Y} = \{1, \dots, g\}$), the class-specific variant of the true positive rate (or rather recall) is

$$\rho_{TPR_i} = \frac{n(i,i)}{n_i}$$

which corresponds to the fraction of correctly classified instances i among all i instances. Thus, the population counterpart is $\mathbb{P}(\hat{y} = i \mid y = i)$, where $\hat{y}$ is the prediction random variable (i.e., $\hat{y} = f(\mathbf{x})$ for a classifier f) and y is the random variable representing the labels.

Positive predictive value:

In the multiclass classification setting ($\mathcal{Y} = \{1, \dots, g\}$), the class-specific variant of the positive predictive value (or rather precision) is

$$\rho_{PPV_i} = \frac{n(i,i)}{\sum_{j=1}^{g} n(i,j)}$$

which corresponds to the fraction of correctly classified instances i among all i classifications. Thus, the population counterpart is $\mathbb{P}(y = i \mid \hat{y} = i)$.

True negative rate:

In the multiclass classification setting ($\mathcal{Y} = \{1, \dots, g\}$), the class-specific variant of the true negative rate is

$$\rho_{TNR_i} = \frac{\sum_{j \neq i} n(j,j)}{n - n_i}$$

which corresponds to the fraction of correctly classified non-i instances among all non-i instances. Thus, the population counterpart is $\mathbb{P}(\hat{y} \neq i \mid y \neq i)$.

Solution 2: PR-curve

Draw the PR-curve for the following data set:

Truth	Score
Pos	0.95
Pos	0.86
Pos	0.69
Neg	0.65
Pos	0.59
Neg	0.52
Pos	0.51
Neg	0.39
Neg	0.28
Neg	0.18
Pos	0.15
Neg	0.06

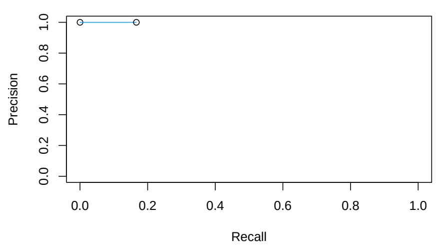
Note that $n_+ = 6 = n_-$ and our classification rule is

$$f(\mathbf{x}) = 2 \cdot \mathbb{1}_{[s(\mathbf{x}) \geq c]} - 1.$$

Step 1: We start with a threshold value of $c = 0.95$, then $TP = 1$, $FP = 0$ and $FN = n_+ - 1$. Thus,

$$\rho_{TPR} = \frac{TP}{TP + FN} = 1/6 \quad \text{and} \quad \rho_{PPV} = \frac{TP}{TP + FP} = 1,$$

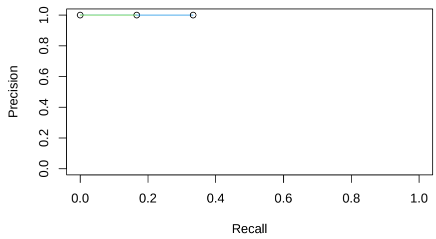
so that we first draw a line from $(0, 1)$ to $(1/6, 1)$.



Step 2: We decrease the threshold value to $c = 0.86$, then $TP = 2$, $FP = 0$ and $FN = n_+ - 2$. Thus,

$$\rho_{TPR} = \frac{TP}{TP + FN} = 2/6 \quad \text{and} \quad \rho_{PPV} = \frac{TP}{TP + FP} = 1,$$

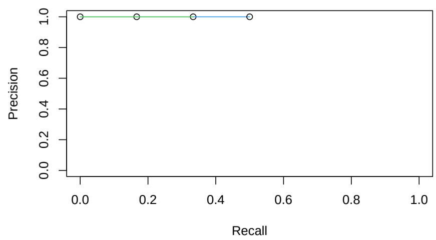
so that we draw another line from $(1/6, 1)$ to $(1/3, 1)$.



Step 3: We decrease the threshold value to $c = 0.69$, then $TP = 3$, $FP = 0$ and $FN = n_+ - 3$. Thus,

$$\rho_{TPR} = \frac{TP}{TP + FN} = 3/6 \quad \text{and} \quad \rho_{PPV} = \frac{TP}{TP + FP} = 1,$$

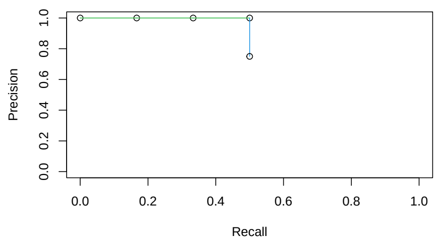
so that we draw another line from $(1/3, 1)$ to $(1/2, 1)$.



Step 4: We decrease the threshold value to $c = 0.65$, then $TP = 3$, $FP = 1$ and $FN = n_+ - 3$. Thus,

$$\rho_{TPR} = \frac{TP}{TP + FN} = 3/6 \quad \text{and} \quad \rho_{PPV} = \frac{TP}{TP + FP} = 3/4,$$

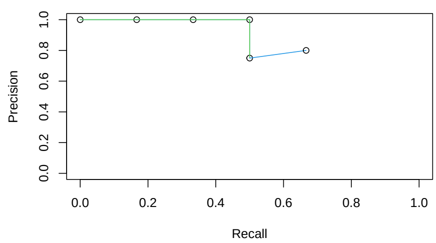
so that we draw another line from $(1/2, 1)$ to $(1/2, 3/4)$.



Step 5: We decrease the threshold value to $c = 0.59$, then $TP = 4$, $FP = 1$ and $FN = n_+ - 4$. Thus,

$$\rho_{TPR} = \frac{TP}{TP + FN} = 4/6 \quad \text{and} \quad \rho_{PPV} = \frac{TP}{TP + FP} = 4/5,$$

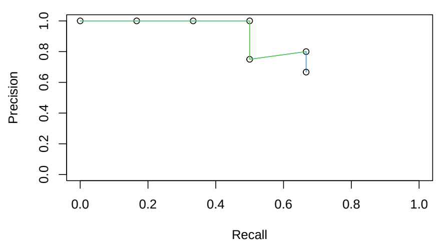
so that we draw another line from $(1/2, 3/4)$ to $(2/3, 4/5)$.



Step 6: We decrease the threshold value to $c = 0.52$, then $TP = 4$, $FP = 2$ and $FN = n_+ - 4$. Thus,

$$\rho_{TPR} = \frac{TP}{TP + FN} = 4/6 \quad \text{and} \quad \rho_{PPV} = \frac{TP}{TP + FP} = 4/6,$$

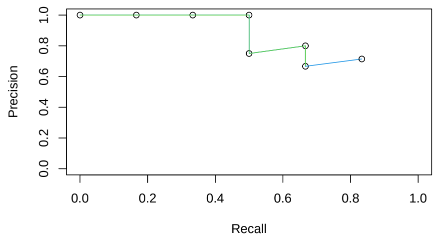
so that we draw another line from $(2/3, 4/5)$ to $(2/3, 2/3)$.



Step 7: We decrease the threshold value to $c = 0.51$, then $TP = 5$, $FP = 2$ and $FN = n_+ - 5$. Thus,

$$\rho_{TPR} = \frac{TP}{TP + FN} = 5/6 \quad \text{and} \quad \rho_{PPV} = \frac{TP}{TP + FP} = 5/7,$$

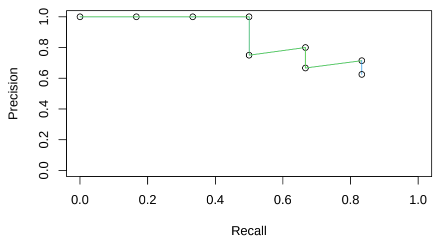
so that we draw another line from $(2/3, 2/3)$ to $(5/6, 5/7)$.



Step 8: We decrease the threshold value to $c = 0.39$, then $TP = 5$, $FP = 3$ and $FN = n_+ - 5$. Thus,

$$\rho_{TPR} = \frac{TP}{TP + FN} = 5/6 \quad \text{and} \quad \rho_{PPV} = \frac{TP}{TP + FP} = 5/8,$$

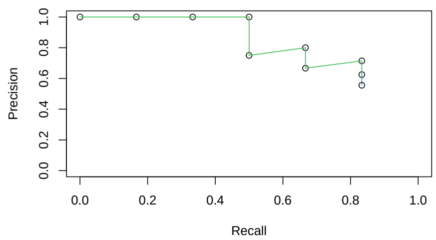
so that we draw another line from $(5/6, 5/7)$ to $(5/6, 5/8)$.



Step 9: We decrease the threshold value to $c = 0.28$, then $TP = 5$, $FP = 4$ and $FN = n_+ - 5$. Thus,

$$\rho_{TPR} = \frac{TP}{TP + FN} = 5/6 \quad \text{and} \quad \rho_{PPV} = \frac{TP}{TP + FP} = 5/9,$$

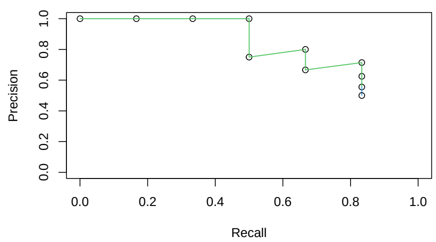
so that we draw another line from $(5/6, 5/8)$ to $(5/6, 5/9)$.



Step 10: We decrease the threshold value to $c = 0.18$, then $TP = 5$, $FP = 5$ and $FN = n_+ - 5$. Thus,

$$\rho_{TPR} = \frac{TP}{TP + FN} = 5/6 \quad \text{and} \quad \rho_{PPV} = \frac{TP}{TP + FP} = 5/10,$$

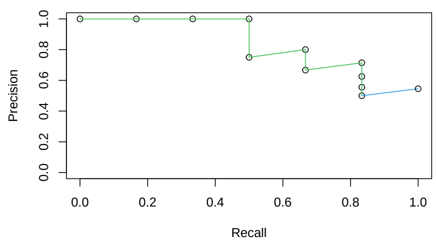
so that we draw another line from $(5/6, 5/9)$ to $(5/6, 1/2)$.



Step 11: We decrease the threshold value to $c = 0.15$, then $TP = 6$, $FP = 5$ and $FN = n_+ - 6 = 0$. Thus,

$$\rho_{TPR} = \frac{TP}{TP + FN} = 1 \quad \text{and} \quad \rho_{PPV} = \frac{TP}{TP + FP} = 6/11,$$

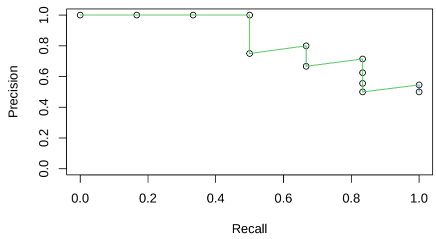
so that we draw another line from $(5/6, 1/2)$ to $(1, 6/11)$.



Step 12: We decrease the threshold value to $c = 0$, then $TP = 6$, $FP = 6$ and $FN = n_+ - 6 = 0$. Thus,

$$\rho_{TPR} = \frac{TP}{TP + FN} = 1 \quad \text{and} \quad \rho_{PPV} = \frac{TP}{TP + FP} = 6/12,$$

so that we draw another line from $(1, 6/11)$ to $(1, 1/2)$.



Solution 3: MetaCost

Implement the MetaCost algorithm and use it with some classifier of your choice on an imbalanced data set of your choice, where the cost-matrix is given by the cost-sensitive heuristic we saw in the lecture. Compare the confusion matrices of the underlying classifier and the MetaCost classifier as well as their total costs.

```r
## --------------------------------------------


library(mlr3)

set.seed(123)


# we use the spam data in the mlr3 package (spam=positive)
task         = tsk("spam")
data         = task$data()
n_vec        = table(data[,1])
g            = length(n_vec) # number of classes (two of course)

# we make a naive training-test split
n            = round(0.8 * task$nrow)
train_set    = 1:n
train_set    = sort(sample(task$nrow, n))
#unique(data[train_set,1])

test_set     = setdiff(seq_len(task$nrow), train_set)

# we use a CART as a classifier
classifier   = "classif.rpart"

# Create the cost matrix via the heuristic

cost_mat = matrix(rep(0,g*g),ncol=2)

for (i in 1:g){
  for (j in 1:g){
    cost_mat[i,j] = max(n_vec[i]/n_vec[j],1)
  }
  cost_mat[i,i]   = 0
}


#  we use CART as the black-box classifier
classifier   = "classif.rpart"
B            = round(n/4)
L            = 10

# 1st phase: Bagging
```

```r
indices     = matrix(rep(0,B*L),ncol=L)

classifiers = c()

for (l in 1:L){

  # bootstrapped data set (or rather the indices)
  indices[,l]          = sort(sample(train_set, size=B, replace = FALSE))
  # fit classifier on bootstrapped data set
  learner              = lrn(classifier, predict_type = "prob")
  classifiers          = c(classifiers,learner$train(task, row_ids = indices[,l]))

}

#    2nd phase: Relabeling

y_tilde          = task$data()[,1]

for (i in train_set){


  # compute the proxy probability vector
  p                = rep(0,g)

  tilde_L          = c()

  # get all bootstrapped data sets, where i is not included

  temp             = which(colSums(i==indices)==0)

  if (length(temp)==0){ # the data point appears in each bootstrapped sample -> use all classifiers
    tilde_L        = 1:L
  } else { # only use classifiers which haven't seen the data point during training
    tilde_L        = sort(temp)
  }


  for (l in tilde_L){
    prediction     = classifiers[[l]]$predict(task, row_ids = i  )
    p              = p + prediction$prob
  }
  # average it
  p                = p/length(tilde_L)

  # relabeling
  # computes the estimated expected costs for predicting class 1,..,g
  est_cost         = cost_mat %*% t(p)
  # assign y the class with lowest estimated expected costs
  y_tilde[i]       = which(est_cost==min(est_cost))[1]

}

# create the relabeled data set
data_rel          =  task$data()
data_rel[,1]      =  y_tilde
task_new          =  TaskClassif$new("rel_data", backend = data_rel, target = task$target_names )

print("Number of relabeled data points")
```

```r
## [1] "Number of relabeled data points"

print(sum(task_new$data()[,1]!=task$data()[,1]))

## [1] 336

# initialize the cost-sensitive classifier
meta_classifier = lrn(classifier, predict_type = "prob")

# 3rd phase: make classifier cost-sensitive
meta_classifier$train(task_new, row_ids = train_set)

# predict with meta

meta_pred   = meta_classifier$predict(task_new, row_ids = test_set)

# train also the cost-insensitive CART
learner     = lrn("classif.rpart", predict_type = "prob")
learner$train(task, row_ids = train_set)
CART_pred   = learner$predict(task, row_ids = test_set)


# Confusion Matrices

meta_pred$confusion

##           truth
## response  spam nonspam
##   spam     316      33
##   nonspam   58     513

CART_pred$confusion

##           truth
## response  spam nonspam
##   spam     308      40
##   nonspam   66     506

# Costs generated

sum(meta_pred$confusion*cost_mat)

## [1] 122.1914

sum(CART_pred$confusion*cost_mat)

## [1] 141.4937
```