

Supplementary Material

This supplementary material includes the required codes for the analysis and study of simulated data and real data. The study was conducted using the free software R.

1. SIMULATIONS

```
# Description: Configuration code for random sample generation (data configuration).
# Necessary to load to run subsequent codes, depending on the number of biomarkers.

configuracion<-c('indep','high cor', 'dif cor', 'neg cor', 'same same','same dif', 'same
dif indep')

## Four biomarkers
mu11<-list(c(0.2,0.5,1.0,0.7), c(0.2,0.5,1.0,0.7), c(0.2,0.5,1.0,0.7), c(0.2,0.5,1.0,0.7)
,
c(1.0,1.0,1.0,1.0), c(1.0,1.0,1.0,1.0), c(1.0,1.0,1.0,1.0))
mu22<-list(c(0.0,0.0,0.0,0.0), c(0.0,0.0,0.0,0.0), c(0.0,0.0,0.0,0.0), c(0.0,0.0,0.0,0.0)
, c(0.0,0.0,0.0,0.0), c(0.0,0.0,0.0,0.0), c(0.0,0.0,0.0,0.0))
Sigma11 <- list(matrix(rbind(c(1.0, 0.0, 0.0, 0.0), c(0.0, 1.0, 0.0, 0.0), c(0.0, 0.0,
1.0, 0.0), c(0.0, 0.0, 0.0, 1.0)),nrow = 4),
matrix(rbind(c(1.0, 0.7, 0.7, 0.7), c(0.7, 1.0, 0.7, 0.7), c(0.7, 0.7, 1.0, 0.7), c(0.7,
0.7, 0.7, 1.0)),nrow = 4),
matrix(rbind(c(1.0, 0.7, 0.7, 0.7), c(0.7, 1.0, 0.7, 0.7), c(0.7, 0.7, 1.0, 0.7), c(0.7,
0.7, 0.7, 1.0)),nrow = 4),
matrix(rbind(c(1.0, -0.1, -0.1,-0.1), c(-0.1, 1.0, -0.1, -0.1), c(-0.1, -0.1, 1.0, -0.1),
c(-0.1, -0.1, -0.1, 1.0)),nrow = 4),
matrix(rbind(c(1.0, 0.3, 0.3, 0.3), c(0.3, 1.0, 0.3, 0.3), c(0.3, 0.3, 1.0, 0.3), c(0.3,
0.3, 0.3, 1.0)),nrow = 4),
matrix(rbind(c(1.0, 0.7, 0.7, 0.7), c(0.7, 1.0, 0.7, 0.7), c(0.7, 0.7, 1.0, 0.7), c(0.7,
0.7, 0.7, 1.0)),nrow = 4),
matrix(rbind(c(1.0, 0.5, 0.5, 0.5), c(0.5, 1.0, 0.5, 0.5), c(0.5, 0.5, 1.0, 0.5), c(0.5,
0.5, 0.5, 1.0)),nrow = 4))
Sigma22 <- list(matrix(rbind(c(1.0, 0.0, 0.0, 0.0), c(0.0, 1.0, 0.0, 0.0), c(0.0, 0.0,
1.0, 0.0), c(0.0, 0.0, 0.0, 1.0)),nrow = 4),
matrix(rbind(c(1.0, 0.7, 0.7, 0.7), c(0.7, 1.0, 0.7, 0.7), c(0.7, 0.7, 1.0, 0.7), c(0.7,
0.7, 0.7, 1.0)),nrow = 4),
matrix(rbind(c(1.0, 0.3, 0.3, 0.3), c(0.3, 1.0, 0.3, 0.3), c(0.3, 0.3, 1.0, 0.3), c(0.3,
0.3, 0.3, 1.0)),nrow = 4),
matrix(rbind(c(1.0, -0.1, -0.1,-0.1), c(-0.1, 1.0, -0.1, -0.1), c(-0.1, -0.1, 1.0, -0.1),
c(-0.1, -0.1, -0.1, 1.0)),nrow = 4),
matrix(rbind(c(1.0, 0.3, 0.3, 0.3), c(0.3, 1.0, 0.3, 0.3), c(0.3, 0.3, 1.0, 0.3), c(0.3,
0.3, 0.3, 1.0)),nrow = 4),
matrix(rbind(c(1.0, 0.3, 0.3, 0.3), c(0.3, 1.0, 0.3, 0.3), c(0.3, 0.3, 1.0, 0.3), c(0.3,
0.3, 0.3, 1.0)),nrow = 4),
matrix(rbind(c(1.0, 0.0, 0.0, 0.0), c(0.0, 1.0, 0.0, 0.0), c(0.0, 0.0, 1.0, 0.0), c(0.0,
0.0, 0.0, 1.0)),nrow = 4))

## Ten biomarkers
mu11<-list(c(0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0), c(0.2, 0.4, 0.6, 0.8,
1.0, 1.2, 1.4, 1.6, 1.8, 2.0),
c(0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0), c(0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4,
1.6, 1.8, 2.0),
c(1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0), c(1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0), c
(1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0))
mu22<-list(c(0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0), c
(0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0),
c(0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0), c(0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0),
c(0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0), c(0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0), c
(0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0))
Sigma11 <- list(matrix(rbind(c(1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0), c(0.0,
1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0), c(0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0), c(0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0), c(0.0, 0.0, 0.0, 0.0,
1.0, 0.0, 0.0, 0.0, 0.0, 0.0),
c(0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0), c(0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
1.0, 0.0, 0.0, 0.0), c(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0), c(0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0),
c(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0)),nrow = 10),
```



```

matrix(rbind(c(1.0, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3), c(0.3, 1.0, 0.3, 0.3,
0.3, 0.3, 0.3, 0.3, 0.3, 0.3), c(0.3, 0.3, 1.0, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3),
c(0.3, 0.3, 0.3, 1.0, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3), c(0.3, 0.3, 0.3, 0.3, 1.0, 0.3,
0.3, 0.3, 0.3, 0.3),
c(0.3, 0.3, 0.3, 0.3, 0.3, 1.0, 0.3, 0.3, 0.3, 0.3), c(0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 1.0,
0.3, 0.3, 0.3),
c(0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 1.0, 0.3, 0.3), c(0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3,
0.3, 1.0, 0.3),
c(0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 1.0)),nrow = 10),
matrix(rbind(c(1.0, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1), c(-0.1, 1.0,
-0.1, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1), c(-0.1, -0.1, 1.0, -0.1, -0.1, -0.1,
-0.1, -0.1, -0.1, -0.1), c(-0.1, -0.1, -0.1, 1.0, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1),
c(-0.1, -0.1, -0.1, -0.1, 1.0, -0.1, -0.1, -0.1, -0.1, -0.1), c(-0.1, -0.1, -0.1, -0.1,
-0.1, 1.0, -0.1, -0.1, -0.1, -0.1),
c(-0.1, -0.1, -0.1, -0.1, -0.1, -0.1, 1.0, -0.1, -0.1, -0.1), c(-0.1, -0.1, -0.1, -0.1,
-0.1, -0.1, -0.1, 1.0, -0.1, -0.1),
c(-0.1, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1, 1.0, -0.1), c(-0.1, -0.1, -0.1, -0.1,
-0.1, -0.1, -0.1, -0.1, -0.1, 1.0)),nrow = 10),
matrix(rbind(c(1.0, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3), c(0.3, 1.0, 0.3, 0.3,
0.3, 0.3, 0.3, 0.3, 0.3, 0.3), c(0.3, 0.3, 1.0, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3),
c(0.3, 0.3, 0.3, 1.0, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3), c(0.3, 0.3, 0.3, 0.3, 1.0, 0.3,
0.3, 0.3, 0.3, 0.3),
c(0.3, 0.3, 0.3, 0.3, 0.3, 1.0, 0.3, 0.3, 0.3, 0.3), c(0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 1.0,
0.3, 0.3, 0.3),
c(0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 1.0, 0.3, 0.3), c(0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3,
0.3, 1.0, 0.3),
c(0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 1.0)),nrow = 10),
matrix(rbind(c(1.0, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3), c(0.3, 1.0, 0.3, 0.3,
0.3, 0.3, 0.3, 0.3, 0.3, 0.3), c(0.3, 0.3, 1.0, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3),
c(0.3, 0.3, 0.3, 1.0, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3), c(0.3, 0.3, 0.3, 0.3, 1.0, 0.3,
0.3, 0.3, 0.3, 0.3),
c(0.3, 0.3, 0.3, 0.3, 0.3, 1.0, 0.3, 0.3, 0.3, 0.3), c(0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 1.0,
0.3, 0.3, 0.3),
c(0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 1.0, 0.3, 0.3), c(0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3,
1.0, 0.3, 0.3),
c(0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 1.0, 0.3), c(0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3,
0.3, 1.0, 0.3),
c(0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 1.0)),nrow = 10),
matrix(rbind(c(1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0), c(0.0, 1.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0), c(0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0),
c(0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0), c(0.0, 0.0, 0.0, 0.0, 1.0, 0.0,
0.0, 0.0, 0.0, 0.0),
c(0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0), c(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0,
0.0, 0.0, 0.0),
c(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0), c(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
1.0, 0.0, 0.0),
c(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0)),nrow = 10))

```

A. Symmetric distributions

A.1. Four biomarkers. Logistic regression

```

# Authors: Rocío Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean and standard deviation of the maximum Youden
# indices obtained through logistic regression on various normal simulated data
# scenarios of four biomarkers.
# Imports needed: MASS
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
# IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
# Youden Index'.

library(MASS)

mean_youden_log<-c()
sd_youden_log<-c()
config<-c()

#Run configuration code for random sample generation#

for (ind in seq(1,length(mu11),1)){

    mu1<-mu11[ind][[1]]
    mu2<-mu22[ind][[1]]

```

```

Sigma1<-Sigma1[ind][[1]]
Sigma2<-Sigma2[ind][[1]]
youden_log<-c()

sequence <- seq(1,100,1)
sequence2<- seq(101,201,1)
n1<-50 #500
n2<-50 #500
count<-1

for(x in sequence){

  ## Training ##
  set.seed(x)
  # create dataset
  DT1<-mvrnorm(n1,mu=mu1,Sigma=Sigma1)
  DT2<-mvrnorm(n2,mu=mu2,Sigma=Sigma2)
  z <- rep(c(1,0), c(n1,n2))
  DT<-data.frame(rbind(DT1,DT2))
  data2<-cbind(DT,z)
  data<-data2
  # train model
  logistic<-glm(z ~ X1+X2+X3+X4, data,family="binomial")
  aa<-unlist(logistic$coefficients[2])*unlist(data[,1])+unlist(logistic$
    coefficients[3])*unlist(data[,2])+unlist(logistic$coefficients[4])*
    unlist(data[,3])+unlist(logistic$coefficients[5])*unlist(data[,4])
  data2<-matrix(aa)
  p<-prediction(data2,data[,dim(data)[2]])
  a<-attributes(performance(p,"sens","spec"))
  cutoff<-a$alpha.values[[1]][which.max(a$y.values[[1]]+a$x.values[[1]]-1)]

  ## Validation ##
  set.seed(sequence2[count])
  # create dataset
  DT1<-mvrnorm(n1,mu=mu1,Sigma=Sigma1)
  DT2<-mvrnorm(n2,mu=mu2,Sigma=Sigma2)
  z <- rep(c(1,0), c(n1,n2))
  DT<-data.frame(rbind(DT1,DT2))
  data2<-cbind(DT,z)
  data<-data2
  # apply model
  aa<-unlist(logistic$coefficients[2])*unlist(data[,1])+unlist(logistic$
    coefficients[3])*unlist(data[,2])+unlist(logistic$coefficients[4])*
    unlist(data[,3])+unlist(logistic$coefficients[5])*unlist(data[,4])
  data22<-matrix(aa)
  data222<-data22[,1]
  data222 <- ifelse(data222<cutoff, 0, 1)
  # calculate Youden index
  p<-prediction(data222,data[,dim(data)[2]])
  at<-attributes(performance(p,"sens","spec"))
  maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
  youden_log<-c(youden_log,maxx)
  count<-count+1

}

config<-c(config,configuracion[ind])
mean_youden_log<-c(mean_youden_log, mean(youden_log))
sd_youden_log<-c(sd_youden_log, sd(youden_log))

}

# Final results
tabla <- data.frame(config, mean_youden_log,sd_youden_log)

```

A.2. Four biomarkers. XGBoost

```

# Authors: Rocío Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean and standard deviation of the maximum Youden
  indices obtained through XGBoost algorithm on various normal simulated data
  scenarios of four biomarkers.

```

```

# Imports needed: MASS, caret, xgboost
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
Youden Index'.

library(MASS)
library(caret)
library(xgboost)

mean_youden_xg<-c()
sd_youden_xg<-c()
config<-c()

youdenSummary <- function(data, lev = NULL, model = NULL){
  if (length(lev) > 2) {
    stop(paste("Your outcome has", length(lev), "levels. The youdenSummary() function isn't
      appropriate."))
  }
  if (!all(levels(data[, "pred"]) == lev)) {
    stop("levels of observed and predicted data do not match")
  }
  Sens <- caret::sensitivity(data[, "pred"], data[, "obs"], lev[1])
  Spec <- caret::specificity(data[, "pred"], data[, "obs"], lev[2])
  j <- Sens + Spec
  out <- c(j, Spec, Sens)
  names(out) <- c("j", "Spec", "Sens")
  out
}

#Run configuration code for random sample generation#

for (ind in seq(1,length(mu11),1)){

  mu1<-mu11[ind][[1]]
  mu2<-mu22[ind][[1]]
  Sigma1<-Sigma11[ind][[1]]
  Sigma2<-Sigma22[ind][[1]]
  youden_xg<-c()

  sequence <- seq(1,100,1)
  sequence2<- seq(101,201,1)
  n1<-50 #500
  n2<-50 #500
  count<-1

  for(x in sequence){

    ## Training ##
    set.seed(x)
    # create dataset
    DT1<-mvrnorm(n1,mu=mu1,Sigma=Sigma1)
    DT2<-mvrnorm(n2,mu=mu2,Sigma=Sigma2)
    z <- rep(c(1,0), c(n1,n2))
    DT<-data.frame(rbind(DT1,DT2))
    data2<-cbind(DT,z)
    data<-data2
    # train model
    data[ , which(names(data) %in% c("z"))]<-as.factor(data[ , which(names(
      data) %in% c("z"))])
    train_control = trainControl(method = "cv", number = 5, search = "grid",
      summaryFunction = youdenSummary)
    gbmGrid <- expand.grid(max_depth = c(2,4,6,10), nrounds = c(50,100,200),
      eta = c(0.1, 0.3), gamma = c(0, 0.5), subsample = c(0.5, 1.0), min_
      child_weight = 1, colsample_bytree = c(0.5,1))
    model0 = train(z~., data = data, method = "xgbTree", metric="j", trControl
      = train_control, tuneGrid = gbmGrid)
    hipers <- model0$finalModel$tuneValue
    data$z<-as.numeric(as.character(data$z))
    model <- xgboost(data = data.matrix(data[ , -which(names(data) %in% c("z"))
      ]), label=as.numeric(data$z), objective = "binary:logistic", early_
      stopping_rounds = 10,nthread = 2, max.depth = hipers$max_depth, eta =
      hipers$eta, nrounds = hipers$nrounds, gamma = hipers$gamma, colsample_

```

```

        bytree = hipers$colsample_bytree, min_child_weight=hipers$min_child_weight, subsample=hipers$subsample)

        dataa<-xgb.DMatrix(data.matrix(data[, -which(names(data) %in% c("z"))]),
            label=data$z)
        predictions <- predict(model, dataa)
        p<-prediction(predictions,data[,dim(data)[2]])
        a<-attributes(performance(p,"sens","spec"))
        cutoff<-a$alpha.values[[1]][which.max(a$y.values[[1]]+a$x.values[[1]]-1)]

        ## Validation ##
        set.seed(sequence2[count])
        # create dataset
        DT1<-mvrnorm(n1,mu=mu1,Sigma=Sigma1)
        DT2<-mvrnorm(n2,mu=mu2,Sigma=Sigma2)
        z <- rep(c(1,0), c(n1,n2))
        DT<-data.frame(rbind(DT1,DT2))
        data2<-cbind(DT,z)
        data<-data2
        # apply model
        dataa<-xgb.DMatrix(data.matrix(data[, -which(names(data) %in% c("z"))]),
            label=data$z)
        predictions <- predict(model, dataa)
        predictions <- ifelse(predictions<cutoff, 0, 1)
        p<-prediction(predictions,data[,dim(data)[2]])
        at<-attributes(performance(p,"sens","spec"))
        maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
        youden_xg<-c(youden_xg,maxx)
        count<-count+1
    }

    config<-c(config,configuracion[ind])
    mean_youden_xg<-c(mean_youden_xg, mean(youden_xg))
    sd_youden_xg<-c(sd_youden_xg, sd(youden_xg))
}

# Final results
tabla <- data.frame(config, mean_youden_xg,sd_youden_xg)

```

A.3. Four biomarkers. Min-max approach

```

# Authors: Rocio Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean and standard deviation of the maximum Youden
               indices obtained through min-max approach on various normal simulated data
               scenarios of four biomarkers.
# Imports needed: MASS, SLModels
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
             IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
             Youden Index', and reference the SLModels library.

library(MASS)
library(SLModels)

mean_youden_mm<-c()
sd_youden_mm<-c()
config<-c()

#Run configuration code for random sample generation#

for (ind in seq(1,length(mu1),1)){

    mu1<-mu1[ind][[1]]
    mu2<-mu2[ind][[1]]
    Sigma1<-Sigma1[ind][[1]]
    Sigma2<-Sigma2[ind][[1]]
    youden_mm<-c()

    sequence <- seq(1,100,1)
    sequence2<- seq(101,201,1)

```

```

n1<-50 #500
n2<-50 #500
count<-1

for(x in sequence){

  ## Training ##
  set.seed(x)
  # create dataset
  DT1<-mvrnorm(n1,mu=mu1,Sigma=Sigma1)
  DT2<-mvrnorm(n2,mu=mu2,Sigma=Sigma2)
  z <- rep(c(1,0), c(n1,n2))
  DT<-data.frame(rbind(DT1,DT2))
  data2<-cbind(DT,z)
  data<-data2
  # train model
  a<-SLModels(data,algorithm="minmax")

  ## Validation ##
  set.seed(sequence2[count])
  # create dataset
  DT1<-mvrnorm(n1,mu=mu1,Sigma=Sigma1)
  DT2<-mvrnorm(n2,mu=mu2,Sigma=Sigma2)
  z <- rep(c(1,0), c(n1,n2))
  DT<-data.frame(rbind(DT1,DT2))
  data2<-cbind(DT,z)
  data<-data2
  # apply model
  aa<-cbind(apply(data[, c(1,2,3,4)],1,max),apply(data[, c(1,2,3,4)],1,min))
  data22<-matrix(unlist(a['max'])*unlist(aa[,1])+unlist(a['min'])*unlist(aa
    [,2]))
  data222<-data22[,1]
  data222 <- ifelse(data222<unlist(a['Cutoff']), 0, 1)
  p<-prediction(data222,data[,dim(data)[2]])
  at<-attributes(performance(p,"sens","spec"))
  maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
  youden_mm<-c(youden_mm,maxx)
  count<-count+1

}

config<-c(config,configuracion[ind])
mean_youden_mm<-c(mean_youden_mm, mean(youden_mm))
sd_youden_mm<-c(sd_youden_mm, sd(youden_mm))

}

# Final results
tabla <- data.frame(config, mean_youden_mm,sd_youden_mm)

```

A.4. Four biomarkers. Min-Max-Median approach

```

# Authors: Rocio Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean and standard deviation of the maximum Youden
  indices obtained through min-max-median approach on various normal simulated data
  scenarios of four biomarkers.
# Imports needed: MASS, SLModels
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
  IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
  Youden Index', and reference the SLModels library.

library(MASS)
library(SLModels)

mean_youden_mmm<-c()
sd_youden_mmm<-c()
config<-c()

#Run configuration code for random sample generation#

for (ind in seq(1,length(mu1),1)){

```

```

mu1<-mu11[ind][[1]]
mu2<-mu22[ind][[1]]
Sigma1<-Sigma11[ind][[1]]
Sigma2<-Sigma22[ind][[1]]
youden_mmm<-c()

sequence <- seq(1,100,1)
sequence2<- seq(101,201,1)
n1<-50 #500
n2<-50 #500
count<-1

for(x in sequence){

  ## Training ##
  set.seed(x)
  # create dataset
  DT1<-mvrnorm(n1,mu=mu1,Sigma=Sigma1)
  DT2<-mvrnorm(n2,mu=mu2,Sigma=Sigma2)
  z <- rep(c(1,0), c(n1,n2))
  DT<-data.frame(rbind(DT1,DT2))
  data2<-cbind(DT,z)
  data<-data2
  # train model
  a<-SLModels(data,algorithm="minmaxmedian")

  ## Validation ##
  set.seed(sequence2[count])
  # create dataset
  DT1<-mvrnorm(n1,mu=mu1,Sigma=Sigma1)
  DT2<-mvrnorm(n2,mu=mu2,Sigma=Sigma2)
  z <- rep(c(1,0), c(n1,n2))
  DT<-data.frame(rbind(DT1,DT2))
  data2<-cbind(DT,z)
  data<-data2
  # apply model
  aa<-cbind(apply(data[, c(1,2,3,4)],1,max),apply(data[, c(1,2,3,4)],1,min),
    apply(data[, c(1,2,3,4)],1,median))
  data22<-matrix(unlist(a['max']))*unlist(aa[,1])+unlist(a['min'])*unlist(aa
    [,2])+unlist(a['median'])*unlist(aa[,3])
  data222<-data22[,1]
  data222 <- ifelse(data222<unlist(a['Cutoff']), 0, 1)
  p<-prediction(data222,data[,dim(data)[2]])
  at<-attributes(performance(p,"sens","spec"))
  maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
  youden_mmm<-c(youden_mmm,maxx)
  count<-count+1

}

config<-c(config,configuracion[ind])
mean_youden_mmm<-c(mean_youden_mmm, mean(youden_mmm))
sd_youden_mmm<-c(sd_youden_mmm, sd(youden_mmm))

}

# Final results
tabla <- data.frame(config, mean_youden_mmm,sd_youden_mmm)

```

A.5. Four biomarkers. Min-Max-IQR approach

```

# Authors: Rocio Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean and standard deviation of the maximum Youden
  indices obtained through min-max-IQR approach on various normal simulated data
  scenarios of four biomarkers.
# Imports needed: MASS, SLModels
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
  IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
  Youden Index', and reference the SLModels library.

```



```

library(MASS)
library(SLModels)

mean_youden_mmiqr<-c()
sd_youden_mmiqr<-c()
config<-c()

#Run configuration code for random sample generation#

for (ind in seq(1,length(mu11),1)){

  mu1<-mu11[ind][[1]]
  mu2<-mu22[ind][[1]]
  Sigma1<-Sigma11[ind][[1]]
  Sigma2<-Sigma22[ind][[1]]
  youden_mmiqr<-c()

  sequence <- seq(1,100,1)
  sequence2<- seq(101,201,1)
  n1<-50 #500
  n2<-50 #500
  count<-1

  for(x in sequence){

    ## Training ##
    set.seed(x)
    # create dataset
    DT1<-mvrnorm(n1,mu=mu1,Sigma=Sigma1)
    DT2<-mvrnorm(n2,mu=mu2,Sigma=Sigma2)
    z <- rep(c(1,0), c(n1,n2))
    DT<-data.frame(rbind(DT1,DT2))
    data2<-cbind(DT,z)
    data<-data2
    # train model
    a<-SLModels(data,algorithm="minmaxiqr")

    ## Validation ##
    set.seed(sequence2[count])
    # create dataset
    DT1<-mvrnorm(n1,mu=mu1,Sigma=Sigma1)
    DT2<-mvrnorm(n2,mu=mu2,Sigma=Sigma2)
    z <- rep(c(1,0), c(n1,n2))
    DT<-data.frame(rbind(DT1,DT2))
    data2<-cbind(DT,z)
    data<-data2
    # apply model
    aa<-cbind(apply(data[,c(1,2,3,4)],1,max),apply(data[,c(1,2,3,4)],1,min),
              apply(data[,c(1,2,3,4)],1,quantile)[3,]-apply(data[,c(1,2,3,4)],1,
              quantile)[1,])
    data22<-matrix(unlist(a['max'])*unlist(aa[,1])+unlist(a['min'])*unlist(aa
    [,2])+unlist(a['iqr'])*unlist(aa[,3]))
    data222<-data22[,1]
    data222 <- ifelse(data222<unlist(a['Cutoff']), 0, 1)
    p<-prediction(data222,data[,dim(data)[2]])
    at<-attributes(performance(p,"sens","spec"))
    maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
    youden_mmiqr<-c(youden_mmiqr,maxx)
    count<-count+1

  }

  config<-c(config,configuracion[ind])
  mean_youden_mmiqr<-c(mean_youden_mmiqr, mean(youden_mmiqr))
  sd_youden_mmiqr<-c(sd_youden_mmiqr, sd(youden_mmiqr))

}

# Final results
tabla <- data.frame(config, mean_youden_mmiqr,sd_youden_mmiqr)

```

A.6. Ten biomarkers. Logistic regression

```
# Authors: Rocio Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean and standard deviation of the maximum Youden
               indices obtained through logistic regression on various normal simulated data
               scenarios of ten biomarkers.
# Imports needed: MASS
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
             IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
             Youden Index'.

library(MASS)

mean_youden_log<-c()
sd_youden_log<-c()
config<-c()

#Run configuration code for random sample generation#

for (ind in seq(1,length(mu1),1)){

  mu1<-mu1[ind][[1]]
  mu2<-mu2[ind][[1]]
  Sigma1<-Sigma1[ind][[1]]
  Sigma2<-Sigma2[ind][[1]]
  youden_log<-c()

  sequence <- seq(1,100,1)
  sequence2<- seq(101,201,1)
  n1<-50 #500
  n2<-50 #500
  count<-1

  for(x in sequence){

    ## Training ##
    set.seed(x)
    # create dataset
    DT1<-mvrnorm(n1,mu=mu1,Sigma=Sigma1)
    DT2<-mvrnorm(n2,mu=mu2,Sigma=Sigma2)
    z <- rep(c(1,0), c(n1,n2))
    DT<-data.frame(rbind(DT1,DT2))
    data2<-cbind(DT,z)
    data<-data2
    # train model
    logistic<-glm(z ~ X1+X2+X3+X4+X5+X6+X7+X8+X9+X10, data,family="binomial")
    aa<-unlist(logistic$coefficients[2])*unlist(data[,1])+unlist(logistic$
      coefficients[3])*unlist(data[,2])+unlist(logistic$coefficients[4])*
      unlist(data[,3])+unlist(logistic$coefficients[5])*unlist(data[,4])+
      unlist(logistic$coefficients[6])*unlist(data[,5])+unlist(logistic$
      coefficients[7])*unlist(data[,6])+unlist(logistic$coefficients[8])*
      unlist(data[,7])+unlist(logistic$coefficients[9])*unlist(data[,8])+
      unlist(logistic$coefficients[10])*unlist(data[,9])+unlist(logistic$
      coefficients[11])*unlist(data[,10])
    data2<-matrix(aa)
    p<-prediction(data2,data[,dim(data)[2]])
    a<-attributes(performance(p,"sens","spec"))
    cutoff<-a$alpha.values[[1]][which.max(a$y.values[[1]]+a$x.values[[1]]-1)]

    ## Validation ##
    set.seed(sequence2[count])
    # create dataset
    DT1<-mvrnorm(n1,mu=mu1,Sigma=Sigma1)
    DT2<-mvrnorm(n2,mu=mu2,Sigma=Sigma2)
    z <- rep(c(1,0), c(n1,n2))
    DT<-data.frame(rbind(DT1,DT2))
    data2<-cbind(DT,z)
    data<-data2
    # apply model
    aa<-unlist(logistic$coefficients[2])*unlist(data[,1])+unlist(logistic$
      coefficients[3])*unlist(data[,2])+unlist(logistic$coefficients[4])*
      unlist(data[,3])+unlist(logistic$coefficients[5])*unlist(data[,4])+
```

```

        unlist(logistic$coefficients[6])*unlist(data[,5])+unlist(logistic$
        coefficients[7])*unlist(data[,6])+unlist(logistic$coefficients[8])*
        unlist(data[,7])+unlist(logistic$coefficients[9])*unlist(data[,8])+
        unlist(logistic$coefficients[10])*unlist(data[,9])+unlist(logistic$
        coefficients[11])*unlist(data[,10])
        data22<-matrix(aa)
        data222<-data22[,1]
        data222 <- ifelse(data222<cutoff, 0, 1)
        # calculate Youden index
        p<-prediction(data222,data[,dim(data)[2]])
        at<-attributes(performance(p,"sens","spec"))
        maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
        youden_log<-c(youden_log,maxx)
        count<-count+1
    }

    config<-c(config,configuracion[ind])
    mean_youden_log<-c(mean_youden_log, mean(youden_log))
    sd_youden_log<-c(sd_youden_log, sd(youden_log))
}

# Final results
tabla <- data.frame(config, mean_youden_log,sd_youden_log)

```

A.7. Ten biomarkers. XGBoost

```

# Authors: Rocio Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean and standard deviation of the maximum Youden
# indices obtained through XGBoost algorithm on various normal simulated data
# scenarios of ten biomarkers.
# Imports needed: MASS, caret, xgboost
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
# IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
# Youden Index'.

library(MASS)
library(caret)
library(xgboost)

mean_youden_xg<-c()
sd_youden_xg<-c()
config<-c()

youdenSummary <- function(data, lev = NULL, model = NULL){
  if (length(lev) > 2) {
    stop(paste("Your outcome has", length(lev), "levels. The youdenSummary() function isn't
    appropriate."))
  }
  if (!all(levels(data[, "pred"]) == lev)) {
    stop("levels of observed and predicted data do not match")
  }
  Sens <- caret::sensitivity(data[, "pred"], data[, "obs"], lev[1])
  Spec <- caret::specificity(data[, "pred"], data[, "obs"], lev[2])
  j <- Sens + Spec
  out <- c(j, Spec, Sens)
  names(out) <- c("j", "Spec", "Sens")
  out
}

#Run configuration code for random sample generation#
for (ind in seq(1,length(mu1),1)){

  mu1<-mu1[ind][[1]]
  mu2<-mu2[ind][[1]]
  Sigma1<-Sigma1[ind][[1]]
  Sigma2<-Sigma2[ind][[1]]
  youden_xg<-c()
}

```

```

sequence <- seq(1,100,1)
sequence2<- seq(101,201,1)
n1<-50 #500
n2<-50 #500
count<-1

for(x in sequence){

  ## Training ##
  set.seed(x)
  # create dataset
  DT1<-mvrnorm(n1,mu=mu1,Sigma=Sigma1)
  DT2<-mvrnorm(n2,mu=mu2,Sigma=Sigma2)
  z <- rep(c(1,0), c(n1,n2))
  DT<-data.frame(rbind(DT1,DT2))
  data2<-cbind(DT,z)
  data<-data2
  # train model
  data[ , which(names(data) %in% c("z"))]<-as.factor(data[ , which(names(
    data) %in% c("z"))])
  train_control = trainControl(method = "cv", number = 5, search = "grid",
    summaryFunction = youdenSummary)
  gbmGrid <- expand.grid(max_depth = c(3,6,8,10,20), nrounds = c(50,100,200),
    eta = c(0.1, 0.3), gamma = c(0, 0.5), subsample = c(0.5, 1.0), min_
    child_weight = 1, colsample_bytree = c(0.5,1))
  model0 = train(z~., data = data, method = "xgbTree", metric="j", trControl
    = train_control, tuneGrid = gbmGrid)
  hipers <- model0$finalModel$tuneValue
  data$z<-as.numeric(as.character(data$z))
  model <- xgboost(data = data.matrix(data[ , -which(names(data) %in% c("z"))
    ]), label=as.numeric(data$z), objective = "binary:logistic", early_
    stopping_rounds = 10,nthread = 2, max.depth = hipers$max_depth, eta =
    hipers$eta, nrounds = hipers$nrounds, gamma = hipers$gamma, colsample_
    bytree = hipers$colsample_bytree, min_child_weight=hipers$min_child_
    weight, subsample=hipers$subsample)

  dataa<-xgb.DMatrix(data.matrix(data[ , -which(names(data) %in% c("z"))]),
    label=data$z)
  predictions <- predict(model, dataa)
  p<-prediction(predictions,data[,dim(data)[2]])
  a<-attributes(performance(p,"sens","spec"))
  cutoff<-a$alpha.values[[1]][which.max(a$y.values[[1]]+a$x.values[[1]]-1)]

  ## Validation ##
  set.seed(sequence2[count])
  # create dataset
  DT1<-mvrnorm(n1,mu=mu1,Sigma=Sigma1)
  DT2<-mvrnorm(n2,mu=mu2,Sigma=Sigma2)
  z <- rep(c(1,0), c(n1,n2))
  DT<-data.frame(rbind(DT1,DT2))
  data2<-cbind(DT,z)
  data<-data2
  # apply model
  dataa<-xgb.DMatrix(data.matrix(data[ , -which(names(data) %in% c("z"))]),
    label=data$z)
  predictions <- predict(model, dataa)
  predictions <- ifelse(predictions<cutoff, 0, 1)
  p<-prediction(predictions,data[,dim(data)[2]])
  at<-attributes(performance(p,"sens","spec"))
  maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
  youden_xg<-c(youden_xg,maxx)
  count<-count+1

}

config<-c(config,configuracion[ind])
mean_youden_xg<-c(mean_youden_xg, mean(youden_xg))
sd_youden_xg<-c(sd_youden_xg, sd(youden_xg))

}

# Final results

```

```
tabla <- data.frame(config, mean_youden_xg, sd_youden_xg)
```

A.8. Ten biomarkers. Min-max approach

```
# Authors: Rocío Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean and standard deviation of the maximum Youden
               indices obtained through min-max approach on various normal simulated data
               scenarios of ten biomarkers.
# Imports needed: MASS, SLModels
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
             IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
             Youden Index', and reference the SLModels library.

library(MASS)
library(SLModels)

mean_youden_mm<-c()
sd_youden_mm<-c()
config<-c()

#Run configuration code for random sample generation#

for (ind in seq(1,length(mu1),1)){

  mu1<-mu1[ind][[1]]
  mu2<-mu2[ind][[1]]
  Sigma1<-Sigma1[ind][[1]]
  Sigma2<-Sigma2[ind][[1]]
  youden_mm<-c()

  sequence <- seq(1,100,1)
  sequence2<- seq(101,201,1)
  n1<-50 #500
  n2<-50 #500
  count<-1

  for(x in sequence){

    ## Training ##
    set.seed(x)
    # create dataset
    DT1<-mvrnorm(n1,mu=mu1,Sigma=Sigma1)
    DT2<-mvrnorm(n2,mu=mu2,Sigma=Sigma2)
    z <- rep(c(1,0), c(n1,n2))
    DT<-data.frame(rbind(DT1,DT2))
    data2<-cbind(DT,z)
    data<-data2
    # train model
    a<-SLModels(data,algorithm="minmax")

    ## Validation ##
    set.seed(sequence2[count])
    # create dataset
    DT1<-mvrnorm(n1,mu=mu1,Sigma=Sigma1)
    DT2<-mvrnorm(n2,mu=mu2,Sigma=Sigma2)
    z <- rep(c(1,0), c(n1,n2))
    DT<-data.frame(rbind(DT1,DT2))
    data2<-cbind(DT,z)
    data<-data2
    # apply model
    aa<-cbind(apply(data[, c(1,2,3,4,5,6,7,8,9,10)],1,max),apply(data[, c
      (1,2,3,4,5,6,7,8,9,10)],1,min))
    data22<-matrix(unlist(a['max'])*unlist(aa[,1])+unlist(a['min'])*unlist(aa
      [,2]))
    data222<-data22[,1]
    data222 <- ifelse(data222<unlist(a['Cutoff']), 0, 1)
    p<-prediction(data222,data[,dim(data)[2]])
    at<-attributes(performance(p,"sens","spec"))
    maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
    youden_mm<-c(youden_mm,maxx)
    count<-count+1
  }
}
```

```

    }

    config<-c(config,configuracion[ind])
    mean_youden_mm<-c(mean_youden_mm, mean(youden_mm))
    sd_youden_mm<-c(sd_youden_mm, sd(youden_mm))
  }

# Final results
tabla <- data.frame(config, mean_youden_mm,sd_youden_mm)

```

A.9. Ten biomarkers. Min-Max-Median approach

```

# Authors: Rocio Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean and standard deviation of the maximum Youden
               indices obtained through min-max-median approach on various normal simulated data
               scenarios of ten biomarkers.
# Imports needed: MASS, SLModels
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
             IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
             Youden Index', and reference the SLModels library.

library(MASS)
library(SLModels)

mean_youden_mmm<-c()
sd_youden_mmm<-c()
config<-c()

#Run configuration code for random sample generation#

for (ind in seq(1,length(mu11),1)){

  mu1<-mu11[ind][[1]]
  mu2<-mu22[ind][[1]]
  Sigma1<-Sigma11[ind][[1]]
  Sigma2<-Sigma22[ind][[1]]
  youden_mmm<-c()

  sequence <- seq(1,100,1)
  sequence2<- seq(101,201,1)
  n1<-50 #500
  n2<-50 #500
  count<-1

  for(x in sequence){

    ## Training ##
    set.seed(x)
    # create dataset
    DT1<-mvrnorm(n1,mu=mu1,Sigma=Sigma1)
    DT2<-mvrnorm(n2,mu=mu2,Sigma=Sigma2)
    z <- rep(c(1,0), c(n1,n2))
    DT<-data.frame(rbind(DT1,DT2))
    data2<-cbind(DT,z)
    data<-data2
    # train model
    a<-SLModels(data,algorithm="minmaxmedian")

    ## Validation ##
    set.seed(sequence2[count])
    # create dataset
    DT1<-mvrnorm(n1,mu=mu1,Sigma=Sigma1)
    DT2<-mvrnorm(n2,mu=mu2,Sigma=Sigma2)
    z <- rep(c(1,0), c(n1,n2))
    DT<-data.frame(rbind(DT1,DT2))
    data2<-cbind(DT,z)
    data<-data2
    # apply model

```

```

aa<-cbind(apply(data[,c(1,2,3,4,5,6,7,8,9,10)],1,max),apply(data[,c
(1,2,3,4,5,6,7,8,9,10)],1,min),apply(data[,c(1,2,3,4,5,6,7,8,9,10)],1,
median))
data22<-matrix(unlist(a['max'])*unlist(aa[,1])+unlist(a['min'])*unlist(aa
[,2])+unlist(a['median'])*unlist(aa[,3]))
data222<-data22[,1]
data222 <- ifelse(data222<unlist(a['Cutoff']), 0, 1)
p<-prediction(data222,data[,dim(data)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
youden_mmm<-c(youden_mmm,maxx)
count<-count+1

}

config<-c(config,configuracion[ind])
mean_youden_mmm<-c(mean_youden_mmm, mean(youden_mmm))
sd_youden_mmm<-c(sd_youden_mmm, sd(youden_mmm))

}

# Final results
tabla <- data.frame(config, mean_youden_mmm,sd_youden_mmm)

```

A.10. Ten biomarkers. Min-Max-IQR approach

```

# Authors: Rocio Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean and standard deviation of the maximum Youden
indices obtained through min-max-IQR approach on various normal simulated data
scenarios of ten biomarkers.
# Imports needed: MASS, SLModels
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
Youden Index', and reference the SLModels library.

library(MASS)
library(SLModels)

mean_youden_mmiqr<-c()
sd_youden_mmiqr<-c()
config<-c()

#Run configuration code for random sample generation#

for (ind in seq(1,length(mu11),1)){

  mu1<-mu11[ind][[1]]
  mu2<-mu22[ind][[1]]
  Sigma1<-Sigma11[ind][[1]]
  Sigma2<-Sigma22[ind][[1]]
  youden_mmiqr<-c()

  sequence <- seq(1,100,1)
  sequence2<- seq(101,201,1)
  n1<-50 #500
  n2<-50 #500
  count<-1

  for(x in sequence){

    ## Training ##
    set.seed(x)
    # create dataset
    DT1<-mvrnorm(n1,mu=mu1,Sigma=Sigma1)
    DT2<-mvrnorm(n2,mu=mu2,Sigma=Sigma2)
    z <- rep(c(1,0), c(n1,n2))
    DT<-data.frame(rbind(DT1,DT2))
    data2<-cbind(DT,z)
    data<-data2
    # train model
    a<-SLModels(data,algorithm="minmaxiqr")
  }
}

```

```

    ## Validation ##
    set.seed(sequence2[count])
    # create dataset
    DT1<-mvrnorm(n1,mu=mu1,Sigma=Sigma1)
    DT2<-mvrnorm(n2,mu=mu2,Sigma=Sigma2)
    z <- rep(c(1,0), c(n1,n2))
    DT<-data.frame(rbind(DT1,DT2))
    data2<-cbind(DT,z)
    data<-data2
    # apply model
    aa<-cbind(apply(data[,c(1,2,3,4,5,6,7,8,9,10)],1,max),apply(data[,c
      (1,2,3,4,5,6,7,8,9,10)],1,min),apply(data[,c(1,2,3,4,5,6,7,8,9,10)],1,
      quantile)[3,]-apply(data[,c(1,2,3,4,5,6,7,8,9,10)],1,quantile)[1,])
    data22<-matrix(unlist(a['max'])*unlist(aa[,1])+unlist(a['min'])*unlist(aa
      [,2])+unlist(a['iqr'])*unlist(aa[,3]))
    data222<-data22[,1]
    data222 <- ifelse(data222<unlist(a['Cutoff']), 0, 1)
    p<-prediction(data222,data[,dim(data)[2]])
    at<-attributes(performance(p,"sens","spec"))
    maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
    youden_mmiqr<-c(youden_mmiqr,maxx)
    count<-count+1
  }

  config<-c(config,configuracion[ind])
  mean_youden_mmiqr<-c(mean_youden_mmiqr, mean(youden_mmiqr))
  sd_youden_mmiqr<-c(sd_youden_mmiqr, sd(youden_mmiqr))
}

# Final results
tabla <- data.frame(config, mean_youden_mmiqr,sd_youden_mmiqr)

```

B. Asymmetric distributions

B.1. Four biomarkers. Log-normal distributions. Logistic regression

```

# Authors: Rocío Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean and standard deviation of the maximum Youden
  indices obtained through logistic regression on various lognormal simulated data
  scenarios of four biomarkers.
# Imports needed: MASS
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
  IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
  Youden Index'.

library(MASS)

mean_youden_log<-c()
sd_youden_log<-c()
config<-c()

#Run configuration code for random sample generation#

for (ind in seq(1,length(mu1),1)){

  mu1<-mu1[ind][[1]]
  mu2<-mu2[ind][[1]]
  Sigma1<-Sigma1[ind][[1]]
  Sigma2<-Sigma2[ind][[1]]
  youden_log<-c()

  sequence <- seq(1,100,1)
  sequence2<- seq(101,201,1)
  n1<-50 #500
  n2<-50 #500
  count<-1

  for(x in sequence){

```



```

## Training ##
set.seed(x)
# create dataset
DT1<-exp(mvrnorm(n1,mu=mu1,Sigma=Sigma1))
DT2<-exp(mvrnorm(n2,mu=mu2,Sigma=Sigma2))
z <- rep(c(1,0), c(n1,n2))
DT<-data.frame(rbind(DT1,DT2))
data2<-cbind(DT,z)
data<-data2
# train model
logistic<-glm(z ~ X1+X2+X3+X4, data,family="binomial")
aa<-unlist(logistic$coefficients[2])*unlist(data[,1])+unlist(logistic$
coefficients[3])*unlist(data[,2])+unlist(logistic$coefficients[4])*
unlist(data[,3])+unlist(logistic$coefficients[5])*unlist(data[,4])
data2<-matrix(aa)
p<-prediction(data2,data[,dim(data)[2]])
a<-attributes(performance(p,"sens","spec"))
cutoff<-a$alpha.values[[1]][which.max(a$y.values[[1]]+a$x.values[[1]]-1)]

## Validation ##
set.seed(sequence2[count])
# create dataset
DT1<-exp(mvrnorm(n1,mu=mu1,Sigma=Sigma1))
DT2<-exp(mvrnorm(n2,mu=mu2,Sigma=Sigma2))
z <- rep(c(1,0), c(n1,n2))
DT<-data.frame(rbind(DT1,DT2))
data2<-cbind(DT,z)
data<-data2
# apply model
aa<-unlist(logistic$coefficients[2])*unlist(data[,1])+unlist(logistic$
coefficients[3])*unlist(data[,2])+unlist(logistic$coefficients[4])*
unlist(data[,3])+unlist(logistic$coefficients[5])*unlist(data[,4])
data22<-matrix(aa)
data222<-data22[,1]
data222 <- ifelse(data222<cutoff, 0, 1)
# calculate Youden index
p<-prediction(data222,data[,dim(data)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
youden_log<-c(youden_log,maxx)
count<-count+1
}

config<-c(config,configuracion[ind])
mean_youden_log<-c(mean_youden_log, mean(youden_log))
sd_youden_log<-c(sd_youden_log, sd(youden_log))
}

# Final results
tabla <- data.frame(config, mean_youden_log,sd_youden_log)

```

B.2. Four biomarkers. Log-normal distributions. XGBoost

```

# Authors: Rocio Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean and standard deviation of the maximum Youden
indices obtained through XGBoost algorithm on various lognormal simulated data
scenarios of four biomarkers.
# Imports needed: MASS, caret, xgboost
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
Youden Index'.

library(MASS)
library(caret)
library(xgboost)

mean_youden_xg<-c()
sd_youden_xg<-c()

```

```

config<-c()

youdenSummary <- function(data, lev = NULL, model = NULL){
  if (length(lev) > 2) {
    stop(paste("Your outcome has", length(lev), "levels. The youdenSummary() function isn't
      appropriate."))
  }
  if (!all(levels(data[, "pred"]) == lev)) {
    stop("levels of observed and predicted data do not match")
  }
  Sens <- caret::sensitivity(data[, "pred"], data[, "obs"], lev[1])
  Spec <- caret::specificity(data[, "pred"], data[, "obs"], lev[2])
  j <- Sens + Spec
  out <- c(j, Spec, Sens)
  names(out) <- c("j", "Spec", "Sens")
  out
}

#Run configuration code for random sample generation#

for (ind in seq(1,length(mu1),1)){

  mu1<-mu1[ind][[1]]
  mu2<-mu2[ind][[1]]
  Sigma1<-Sigma1[ind][[1]]
  Sigma2<-Sigma2[ind][[1]]
  youden_xg<-c()

  sequence <- seq(1,100,1)
  sequence2<- seq(101,201,1)
  n1<-50 #500
  n2<-50 #500
  count<-1

  for(x in sequence){

    ## Training ##
    set.seed(x)
    # create dataset
    DT1<-exp(mvrnorm(n1,mu=mu1,Sigma=Sigma1))
    DT2<-exp(mvrnorm(n2,mu=mu2,Sigma=Sigma2))
    z <- rep(c(1,0), c(n1,n2))
    DT<-data.frame(rbind(DT1,DT2))
    data2<-cbind(DT,z)
    data<-data2
    # train model
    data[, which(names(data) %in% c("z"))]<-as.factor(data[, which(names(
      data) %in% c("z"))])
    train_control = trainControl(method = "cv", number = 5, search = "grid",
      summaryFunction = youdenSummary)
    gbmGrid <- expand.grid(max_depth = c(2,4,6,10), nrounds = c(50,100,200),
      eta = c(0.1, 0.3), gamma = c(0, 0.5), subsample = c(0.5, 1.0), min_
      child_weight = 1, colsample_bytree = c(0.5,1))
    model0 = train(z~., data = data, method = "xgbTree", metric="j", trControl
      = train_control, tuneGrid = gbmGrid)
    hipers <- model0$finalModel$tuneValue
    data$z<-as.numeric(as.character(data$z))
    model <- xgboost(data = data.matrix(data[, -which(names(data) %in% c("z"))
      ]), label=as.numeric(data$z), objective = "binary:logistic", early_
      stopping_rounds = 10,nthread = 2, max.depth = hipers$max_depth, eta =
      hipers$eta, nrounds = hipers$nrounds, gamma = hipers$gamma, colsample_
      bytree = hipers$colsample_bytree, min_child_weight=hipers$min_child_
      weight, subsample=hipers$subsample)

    dataa<-xgb.DMatrix(data.matrix(data[, -which(names(data) %in% c("z"))]),
      label=data$z)
    predictions <- predict(model, dataa)
    p<-prediction(predictions,data[,dim(data)[2]])
    a<-attributes(performance(p,"sens","spec"))
    cutoff<-a$alpha.values[[1]][which.max(a$y.values[[1]]+a$x.values[[1]]-1)]

    ## Validation ##
  }
}

```

```

        set.seed(sequence2[count])
        # create dataset
        DT1<-exp(mvrnorm(n1,mu=mu1,Sigma=Sigma1))
        DT2<-exp(mvrnorm(n2,mu=mu2,Sigma=Sigma2))
        z <- rep(c(1,0), c(n1,n2))
        DT<-data.frame(rbind(DT1,DT2))
        data2<-cbind(DT,z)
        data<-data2
        # apply model
        dataa<-xgb.DMatrix(data.matrix(data[, -which(names(data) %in% c("z"))]),
            label=data$z)
        predictions <- predict(model, dataa)
        predictions <- ifelse(predictions<cutoff, 0, 1)
        p<-prediction(predictions,data[,dim(data)[2]])
        at<-attributes(performance(p,"sens","spec"))
        maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
        youden_xg<-c(youden_xg,maxx)
        count<-count+1
    }

    config<-c(config,configuracion[ind])
    mean_youden_xg<-c(mean_youden_xg, mean(youden_xg))
    sd_youden_xg<-c(sd_youden_xg, sd(youden_xg))
}

# Final results
tabla <- data.frame(config, mean_youden_xg,sd_youden_xg)

```

B.3. Four biomarkers. Log-normal distributions. Min-max approach

```

# Authors: Rocio Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean and standard deviation of the maximum Youden
              indices obtained through min-max approach on various lognormal simulated data
              scenarios of four biomarkers.
# Imports needed: MASS, SLModels
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
              IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
              Youden Index', and reference the SLModels library.

library(MASS)
library(SLModels)

mean_youden_mm<-c()
sd_youden_mm<-c()
config<-c()

#Run configuration code for random sample generation#

for (ind in seq(1,length(mu11),1)){

    mu1<-mu11[ind][[1]]
    mu2<-mu22[ind][[1]]
    Sigma1<-Sigma11[ind][[1]]
    Sigma2<-Sigma22[ind][[1]]
    youden_mm<-c()

    sequence <- seq(1,100,1)
    sequence2<- seq(101,201,1)
    n1<-50 #500
    n2<-50 #500
    count<-1

    for(x in sequence){

        ## Training ##
        set.seed(x)
        # create dataset
        DT1<-exp(mvrnorm(n1,mu=mu1,Sigma=Sigma1))
        DT2<-exp(mvrnorm(n2,mu=mu2,Sigma=Sigma2))
    }
}

```

```

z <- rep(c(1,0), c(n1,n2))
DT<-data.frame(rbind(DT1,DT2))
data2<-cbind(DT,z)
data<-data2
# train model
a<-SLModels(data,algorithm="minmax")

## Validation ##
set.seed(sequence2[count])
# create dataset
DT1<-exp(mvrnorm(n1,mu=mu1,Sigma=Sigma1))
DT2<-exp(mvrnorm(n2,mu=mu2,Sigma=Sigma2))
z <- rep(c(1,0), c(n1,n2))
DT<-data.frame(rbind(DT1,DT2))
data2<-cbind(DT,z)
data<-data2
# apply model
aa<-cbind(apply(data[, c(1,2,3,4)],1,max),apply(data[, c(1,2,3,4)],1,min))
data22<-matrix(unlist(a['max'])*unlist(aa[,1])+unlist(a['min'])*unlist(aa
[,2]))
data222<-data22[,1]
data222 <- ifelse(data222<unlist(a['Cutoff']), 0, 1)
p<-prediction(data222,data[,dim(data)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
youden_mm<-c(youden_mm,maxx)
count<-count+1

}

config<-c(config,configuracion[ind])
mean_youden_mm<-c(mean_youden_mm, mean(youden_mm))
sd_youden_mm<-c(sd_youden_mm, sd(youden_mm))

}

# Final results
tabla <- data.frame(config, mean_youden_mm,sd_youden_mm)

```

B.4. Four biomarkers. Log-normal distributions. Min-Max-Median approach

```

# Authors: Rocio Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean and standard deviation of the maximum Youden
indices obtained through min-max-median approach on various lognormal simulated
data scenarios of four biomarkers.
# Imports needed: MASS, SLModels
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
Youden Index', and reference the SLModels library.

library(MASS)
library(SLModels)

mean_youden_mmm<-c()
sd_youden_mmm<-c()
config<-c()

#Run configuration code for random sample generation#
for (ind in seq(1,length(mu1),1)){

  mu1<-mu1[ind][[1]]
  mu2<-mu2[ind][[1]]
  Sigma1<-Sigma1[ind][[1]]
  Sigma2<-Sigma2[ind][[1]]
  youden_mmm<-c()

  sequence <- seq(1,100,1)
  sequence2<- seq(101,201,1)
  n1<-50 #500
  n2<-50 #500

```

```

count<-1

for(x in sequence){

  ## Training ##
  set.seed(x)
  # create dataset
  DT1<-exp(mvrnorm(n1,mu=mu1,Sigma=Sigma1))
  DT2<-exp(mvrnorm(n2,mu=mu2,Sigma=Sigma2))
  z <- rep(c(1,0), c(n1,n2))
  DT<-data.frame(rbind(DT1,DT2))
  data2<-cbind(DT,z)
  data<-data2
  # train model
  a<-SLModels(data,algorithm="minmaxmedian")

  ## Validation ##
  set.seed(sequence2[count])
  # create dataset
  DT1<-exp(mvrnorm(n1,mu=mu1,Sigma=Sigma1))
  DT2<-exp(mvrnorm(n2,mu=mu2,Sigma=Sigma2))
  z <- rep(c(1,0), c(n1,n2))
  DT<-data.frame(rbind(DT1,DT2))
  data2<-cbind(DT,z)
  data<-data2
  # apply model
  aa<-cbind(apply(data[, c(1,2,3,4)],1,max),apply(data[, c(1,2,3,4)],1,min),
            apply(data[, c(1,2,3,4)],1,median))
  data22<-matrix(unlist(a['max'])*unlist(aa[,1])+unlist(a['min'])*unlist(aa
            [,2])+unlist(a['median'])*unlist(aa[,3]))
  data222<-data22[,1]
  data222 <- ifelse(data222<unlist(a['Cutoff']), 0, 1)
  p<-prediction(data222,data[,dim(data)[2]])
  at<-attributes(performance(p,"sens","spec"))
  maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
  youden_mmm<-c(youden_mmm,maxx)
  count<-count+1

}

config<-c(config,configuracion[ind])
mean_youden_mmm<-c(mean_youden_mmm, mean(youden_mmm))
sd_youden_mmm<-c(sd_youden_mmm, sd(youden_mmm))

}

# Final results
tabla <- data.frame(config, mean_youden_mmm,sd_youden_mmm)

```

B.5. Four biomarkers. Log-normal distributions. Min-Max-IQR approach

```

# Authors: Rocío Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean and standard deviation of the maximum Youden
               indices obtained through min-max-IQR approach on various lognormal simulated data
               scenarios of four biomarkers.
# Imports needed: MASS, SLModels
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
             IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
             Youden Index', and reference the SLModels library.

library(MASS)
library(SLModels)

mean_youden_mmiqr<-c()
sd_youden_mmiqr<-c()
config<-c()

#Run configuration code for random sample generation#

for (ind in seq(1,length(mu1),1)){

```

```

mu1<-mu1[ind][[1]]
mu2<-mu2[ind][[1]]
Sigma1<-Sigma1[ind][[1]]
Sigma2<-Sigma2[ind][[1]]
youden_mmiqr<-c()

sequence <- seq(1,100,1)
sequence2<- seq(101,201,1)
n1<-50 #500
n2<-50 #500
count<-1

for(x in sequence){

  ## Training ##
  set.seed(x)
  # create dataset
  DT1<-exp(mvrnorm(n1,mu=mu1,Sigma=Sigma1))
  DT2<-exp(mvrnorm(n2,mu=mu2,Sigma=Sigma2))
  z <- rep(c(1,0), c(n1,n2))
  DT<-data.frame(rbind(DT1,DT2))
  data2<-cbind(DT,z)
  data<-data2
  # train model
  a<-SLModels(data,algorithm="minmaxiqr")

  ## Validation ##
  set.seed(sequence2[count])
  # create dataset
  DT1<-exp(mvrnorm(n1,mu=mu1,Sigma=Sigma1))
  DT2<-exp(mvrnorm(n2,mu=mu2,Sigma=Sigma2))
  z <- rep(c(1,0), c(n1,n2))
  DT<-data.frame(rbind(DT1,DT2))
  data2<-cbind(DT,z)
  data<-data2
  # apply model
  aa<-cbind(apply(data[,c(1,2,3,4)],1,max),apply(data[,c(1,2,3,4)],1,min),
            apply(data[,c(1,2,3,4)],1,quantile)[3,]-apply(data[,c(1,2,3,4)],1,
            quantile)[1,])
  data22<-matrix(unlist(a['max'])*unlist(aa[,1])+unlist(a['min'])*unlist(aa
            [,2])+unlist(a['iqr'])*unlist(aa[,3]))
  data222<-data22[,1]
  data222 <- ifelse(data222<unlist(a['Cutoff']), 0, 1)
  p<-prediction(data222,data[,dim(data)[2]])
  at<-attributes(performance(p,"sens","spec"))
  maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
  youden_mmiqr<-c(youden_mmiqr,maxx)
  count<-count+1

}

config<-c(config,configuracion[ind])
mean_youden_mmiqr<-c(mean_youden_mmiqr, mean(youden_mmiqr))
sd_youden_mmiqr<-c(sd_youden_mmiqr, sd(youden_mmiqr))

}

# Final results
tabla <- data.frame(config, mean_youden_mmiqr,sd_youden_mmiqr)

```

B.6. Four biomarkers. Different marginal distributions

```

# Authors: Rocio Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean and standard deviation of the maximum Youden
              indices obtained through logistic regression, XGBoost algorithm, min-max approach,
              min-max-median approach and min-max-IQR, considering different marginal
              distributions (chi2, normal, gamma and exponential).
# Imports needed: MASS, SLModels, caret, xgboost, copula, coop
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
              IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
              Youden Index', and reference the SLModels library.

```

```

library(MASS)
library(SLModels)
library(caret)
library(xgboost)
library(copula)
library(coop)

configuracion<-c('marg')
youden_log<-c()
youden_xg<-c()
youden_mm<-c()
youden_mmm<-c()
youden_mmiqr<-c()

sequence <- seq(1,100,1)
sequence2<- seq(101,201,1)
n1<-50 #500
n2<-50 #500
count<-1

youdenSumary <- function(data, lev = NULL, model = NULL){
  if (length(lev) > 2) {
    stop(paste("Your outcome has", length(lev), "levels. The youdenSumary() function isn't
      appropriate."))
  }
  if (!all(levels(data[, "pred"]) == lev)) {
    stop("levels of observed and predicted data do not match")
  }
  Sens <- caret::sensitivity(data[, "pred"], data[, "obs"], lev[1])
  Spec <- caret::specificity(data[, "pred"], data[, "obs"], lev[2])
  j <- Sens + Spec
  out <- c(j, Spec, Sens)
  names(out) <- c("j", "Spec", "Sens")
  out
}

for(x in sequence){

  ## Training ##
  set.seed(x)
  # create dataset
  sanos <- mvdc(normalCopula(0.3, dim=4), c("chisq", "norm", "gamma", "exp"),
    list(list(0.1), list(mean = 0.1, sd =1), list(shape=0.1, rate=1), list(rate = 0.1)
    ))
  enfermos <- mvdc(normalCopula(0.7, dim=4), c("chisq", "norm", "gamma", "exp"),
    list(list(0.1), list(mean = 0.6, sd =1), list(shape=0.8, rate=1), list(rate = 0.1)
    ))
  DT2<-rMvdc(n2, sanos)
  DT1<-rMvdc(n1, enfermos)
  z <- rep(c(1,0), c(n1,n2))
  DT<-data.frame(rbind(DT1,DT2))
  data2<-cbind(DT,z)
  data0<-data2
  data<-data2

  ## Logistic regression
  logistic<-glm(z ~ X1+X2+X3+X4, data,family="binomial")
  aa<-unlist(logistic$coefficients[2])*unlist(data[,1])+unlist(logistic$coefficients
    [3])*unlist(data[,2])+unlist(logistic$coefficients[4])*unlist(data[,3])+
    unlist(logistic$coefficients[5])*unlist(data[,4])
  data2<-matrix(aa)
  p<-prediction(data2,data[,dim(data)[2]])
  a<-attributes(performance(p,"sens","spec"))
  cutofflog<-a$alpha.values[[1]][which.max(a$y.values[[1]]+a$x.values[[1]]-1)]
  ## XGBoost
  data[, which(names(data) %in% c("z"))]<-as.factor(data[, which(names(data) %in%
    c("z"))])
  train_control = trainControl(method = "cv", number = 5, search = "grid",
    summaryFunction = youdenSumary)
  gbmGrid <- expand.grid(max_depth = c(2,4,6,10), nrounds = c(50,100,200), eta = c
    (0.1, 0.3), gamma = c(0, 0.5), subsample = c(0.5, 1.0), min_child_weight = 1,

```

```

      colsample_bytree = c(0.5,1))
model0 = train(z~., data = data, method = "xgbTree", metric="j", trControl = train
  _control, tuneGrid = gbmGrid)
hipers <- model0$finalModel$tuneValue
data$z<-as.numeric(as.character(data$z))
model <- xgboost(data = data.matrix(data[ , -which(names(data) %in% c("z"))]),
  label=as.numeric(data$z), objective = "binary:logistic", max.depth = hipers$
  max_depth, eta = hipers$eta, nthread = 2, nrounds = hipers$nrounds, early_
  stopping_rounds = 10, gamma = hipers$gamma, colsample_bytree = hipers$
  colsample_bytree, min_child_weight=hipers$min_child_weight, subsample=hipers$
  subsample)
dataa<-xgb.DMatrix(data.matrix(data[ , -which(names(data) %in% c("z"))]), label=
  data$z)
predictions <- predict(model, dataa)
p<-prediction(predictions,data[,dim(data)[2]])
a<-attributes(performance(p,"sens","spec"))
cutoffxg<-a$alpha.values[[1]][which.max(a$y.values[[1]]+a$x.values[[1]]-1)]
## MM
data<-data0
data$X1<-scaler(data$X1)
data$X2<-scaler(data$X2)
data$X3<-scaler(data$X3)
data$X4<-scaler(data$X4)
a_mm<-SLModels(data,algorithm="minmax")
## MMM
a_mmm<-SLModels(data,algorithm="minmaxmedian")
## MMIQR
a_mmiqr<-SLModels(data,algorithm="minmaxiqr")

## Validation ##
set.seed(sequence2[count])
# create dataset
sanos <- mvdc(normalCopula(0.3, dim=4), c("chisq", "norm", "gamma", "exp"),
  list(list(0.1), list(mean = 0.1, sd =1), list(shape=0.1, rate=1), list(rate = 0.1)
  ))
enfermos <- mvdc(normalCopula(0.7, dim=4), c("chisq", "norm", "gamma", "exp"),
  list(list(0.1), list(mean = 0.6, sd =1), list(shape=0.8, rate=1), list(rate = 0.1)
  ))
DT2<-rMvdc(n2, sanos)
DT1<-rMvdc(n1, enfermos)
z <- rep(c(1,0), c(n1,n2))
DT<-data.frame(rbind(DT1,DT2))
data2<-cbind(DT,z)
data0<-data2
data<-data2

## Logistic regression
aa<-unlist(logistic$coefficients[2])*unlist(data[,1])+unlist(logistic$coefficients
  [3])*unlist(data[,2])+unlist(logistic$coefficients[4])*unlist(data[,3])+
  unlist(logistic$coefficients[5])*unlist(data[,4])
data22<-matrix(aa)
data222<-data22[,1]
data222 <- ifelse(data222<cutofflog, 0, 1)
p<-prediction(data222,data[,dim(data)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
youden_log<-c(youden_log,maxx)
## XGBoost
dataa<-xgb.DMatrix(data.matrix(data[ , -which(names(data) %in% c("z"))]), label=
  data$z)
predictions <- predict(model, dataa)
predictions <- ifelse(predictions<cutoffxg, 0, 1)
p<-prediction(predictions,data[,dim(data)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
youden_xg<-c(youden_xg,maxx)
## MM
data<-data0
data$X1<-scaler(data$X1)
data$X2<-scaler(data$X2)
data$X3<-scaler(data$X3)
data$X4<-scaler(data$X4)

```



```

aa<-cbind(apply(data[, c(1,2,3,4)],1,max),apply(data[, c(1,2,3,4)],1,min))
data22<-matrix(unlist(a_mm['max'])*unlist(aa[,1])+unlist(a_mm['min'])*unlist(aa
[,2]))
data222<-data22[,1]
data222 <- ifelse(data222<unlist(a_mm['Cutoff']), 0, 1)
p<-prediction(data222,data[,dim(data)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
youden_mm<-c(youden_mm,maxx)
## MMM
aa<-cbind(apply(data[, c(1,2,3,4)],1,max),apply(data[, c(1,2,3,4)],1,min),apply(
data[, c(1,2,3,4)],1,median))
data22<-matrix(unlist(a_mmm['max'])*unlist(aa[,1])+unlist(a_mmm['min'])*unlist(aa
[,2])+unlist(a_mmm['median'])*unlist(aa[,3]))
data222<-data22[,1]
data222 <- ifelse(data222<unlist(a_mmm['Cutoff']), 0, 1)
p<-prediction(data222,data[,dim(data)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
youden_mmm<-c(youden_mmm,maxx)
## MMIQR
aa<-cbind(apply(data[, c(1,2,3,4)],1,max),apply(data[, c(1,2,3,4)],1,min),apply(
data[, c(1,2,3,4)],1,quantile)[3,]-apply(data[, c(1,2,3,4)],1,quantile)[1,])
data22<-matrix(unlist(a_mmiqr['max'])*unlist(aa[,1])+unlist(a_mmiqr['min'])*unlist
(aa[,2])+unlist(a_mmiqr['iqr'])*unlist(aa[,3]))
data222<-data22[,1]
data222 <- ifelse(data222<unlist(a_mmiqr['Cutoff']), 0, 1)
p<-prediction(data222,data[,dim(data)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
youden_mmiqr<-c(youden_mmiqr,maxx)

count<-count+1
}

# Final results
config<-c("logistic", "xgboost","mm", "mmm", "mmiqr")
mean_youden_log<-c(mean(youden_log),mean(youden_xg), mean(youden_mm), mean(youden_mmm),
mean(youden_mmiqr))
sd_youden_log<-c(sd(youden_log), sd(youden_xg), sd(youden_mm), sd(youden_mmm), sd(youden_
mmiqr))
tabla <- data.frame(config, mean_youden_log,sd_youden_log,mean_auc_log,sd_auc_log)

```

B.7. Ten biomarkers. Log-normal distributions. Logistic regression

```

# Authors: Rocio Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean and standard deviation of the maximum Youden
indices obtained through logistic regression on various lognormal simulated data
scenarios of ten biomarkers.
# Imports needed: MASS
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
Youden Index'.

library(MASS)

mean_youden_log<-c()
sd_youden_log<-c()
config<-c()

#Run configuration code for random sample generation#

for (ind in seq(1,length(mu11),1)){

  mu1<-mu11[ind][[1]]
  mu2<-mu22[ind][[1]]
  Sigma1<-Sigma11[ind][[1]]
  Sigma2<-Sigma22[ind][[1]]
  youden_log<-c()

  sequence <- seq(1,100,1)

```

```

sequence2<- seq(101,201,1)
n1<-50 #500
n2<-50 #500
count<-1

for(x in sequence){

  ## Training ##
  set.seed(x)
  # create dataset
  DT1<-exp(mvrnorm(n1,mu=mu1,Sigma=Sigma1))
  DT2<-exp(mvrnorm(n2,mu=mu2,Sigma=Sigma2))
  z <- rep(c(1,0), c(n1,n2))
  DT<-data.frame(rbind(DT1,DT2))
  data2<-cbind(DT,z)
  data<-data2
  # train model
  logistic<-glm(z ~ X1+X2+X3+X4+X5+X6+X7+X8+X9+X10, data,family="binomial")
  aa<-unlist(logistic$coefficients[2])*unlist(data[,1])+unlist(logistic$
    coefficients[3])*unlist(data[,2])+unlist(logistic$coefficients[4])*
    unlist(data[,3])+unlist(logistic$coefficients[5])*unlist(data[,4])+
    unlist(logistic$coefficients[6])*unlist(data[,5])+unlist(logistic$
    coefficients[7])*unlist(data[,6])+unlist(logistic$coefficients[8])*
    unlist(data[,7])+unlist(logistic$coefficients[9])*unlist(data[,8])+
    unlist(logistic$coefficients[10])*unlist(data[,9])+unlist(logistic$
    coefficients[11])*unlist(data[,10])
  data2<-matrix(aa)
  p<-prediction(data2,data[,dim(data)[2]])
  a<-attributes(performance(p,"sens","spec"))
  cutoff<-a$alpha.values[[1]][which.max(a$y.values[[1]]+a$x.values[[1]]-1)]

  ## Validation ##
  set.seed(sequence2[count])
  # create dataset
  DT1<-exp(mvrnorm(n1,mu=mu1,Sigma=Sigma1))
  DT2<-exp(mvrnorm(n2,mu=mu2,Sigma=Sigma2))
  z <- rep(c(1,0), c(n1,n2))
  DT<-data.frame(rbind(DT1,DT2))
  data2<-cbind(DT,z)
  data<-data2
  # apply model
  aa<-unlist(logistic$coefficients[2])*unlist(data[,1])+unlist(logistic$
    coefficients[3])*unlist(data[,2])+unlist(logistic$coefficients[4])*
    unlist(data[,3])+unlist(logistic$coefficients[5])*unlist(data[,4])+
    unlist(logistic$coefficients[6])*unlist(data[,5])+unlist(logistic$
    coefficients[7])*unlist(data[,6])+unlist(logistic$coefficients[8])*
    unlist(data[,7])+unlist(logistic$coefficients[9])*unlist(data[,8])+
    unlist(logistic$coefficients[10])*unlist(data[,9])+unlist(logistic$
    coefficients[11])*unlist(data[,10])
  data22<-matrix(aa)
  data222<-data22[,1]
  data222 <- ifelse(data222<cutoff, 0, 1)
  # calculate Youden index
  p<-prediction(data222,data[,dim(data)[2]])
  at<-attributes(performance(p,"sens","spec"))
  maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
  youden_log<-c(youden_log,maxx)
  count<-count+1

}

config<-c(config,configuracion[ind])
mean_youden_log<-c(mean_youden_log, mean(youden_log))
sd_youden_log<-c(sd_youden_log, sd(youden_log))

}

# Final results
tabla <- data.frame(config, mean_youden_log,sd_youden_log)

```

B.8. Ten biomarkers. Log-normal distributions. XGBoost

```

# Authors: Rocio Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean and standard deviation of the maximum Youden
               indices obtained through XGBoost algorithm on various lognormal simulated data
               scenarios of ten biomarkers.
# Imports needed: MASS, caret, xgboost
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
             IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
             Youden Index'.

library(MASS)
library(caret)
library(xgboost)

mean_youden_xg<-c()
sd_youden_xg<-c()
config<-c()

youdenSummary <- function(data, lev = NULL, model = NULL){
  if (length(lev) > 2) {
    stop(paste("Your outcome has", length(lev), "levels. The youdenSummary() function isn't
               appropriate."))
  }
  if (!all(levels(data[, "pred"]) == lev)) {
    stop("levels of observed and predicted data do not match")
  }
  Sens <- caret::sensitivity(data[, "pred"], data[, "obs"], lev[1])
  Spec <- caret::specificity(data[, "pred"], data[, "obs"], lev[2])
  j <- Sens + Spec
  out <- c(j, Spec, Sens)
  names(out) <- c("j", "Spec", "Sens")
  out
}

#Run configuration code for random sample generation#

for (ind in seq(1,length(mu11),1)){

  mu1<-mu11[ind][[1]]
  mu2<-mu22[ind][[1]]
  Sigma1<-Sigma11[ind][[1]]
  Sigma2<-Sigma22[ind][[1]]
  youden_xg<-c()

  sequence <- seq(1,100,1)
  sequence2<- seq(101,201,1)
  n1<-50 #500
  n2<-50 #500
  count<-1

  for(x in sequence){

    ## Training ##
    set.seed(x)
    # create dataset
    DT1<-exp(mvrnorm(n1,mu=mu1,Sigma=Sigma1))
    DT2<-exp(mvrnorm(n2,mu=mu2,Sigma=Sigma2))
    z <- rep(c(1,0), c(n1,n2))
    DT<-data.frame(rbind(DT1,DT2))
    data2<-cbind(DT,z)
    data<-data2
    # train model
    data[ , which(names(data) %in% c("z"))]<-as.factor(data[ , which(names(
      data) %in% c("z"))])
    train_control = trainControl(method = "cv", number = 5, search = "grid",
      summaryFunction = youdenSummary)
    gbmGrid <- expand.grid(max_depth = c(3,6,8,10,20), nrounds = c(50,100,200),
      eta = c(0.1, 0.3), gamma = c(0, 0.5), subsample = c(0.5, 1.0), min_
      child_weight = 1, colsample_bytree = c(0.5,1))
    model0 = train(z~., data = data, method = "xgbTree", metric="j", trControl
      = train_control, tuneGrid = gbmGrid)
    hipers <- model0$finalModel$tuneValue
  }
}

```

```

data$z<-as.numeric(as.character(data$z))
model <- xgboost(data = data.matrix(data[, -which(names(data) %in% c("z"))
]), label=as.numeric(data$z), objective = "binary:logistic", early_
stopping_rounds = 10,nthread = 2, max.depth = hipers$max_depth, eta =
hipers$eta, nrounds = hipers$nrounds, gamma = hipers$gamma, colsample_
bytree = hipers$colsample_bytree, min_child_weight=hipers$min_child_
weight, subsample=hipers$subsample)

dataa<-xgb.DMatrix(data.matrix(data[, -which(names(data) %in% c("z"))]),
label=data$z)
predictions <- predict(model, dataa)
p<-prediction(predictions,data[,dim(data)[2]])
a<-attributes(performance(p,"sens","spec"))
cutoff<-a$alpha.values[[1]][which.max(a$y.values[[1]]+a$x.values[[1]]-1)]

## Validation ##
set.seed(sequence2[count])
# create dataset
DT1<-exp(mvrnorm(n1,mu=mu1,Sigma=Sigma1))
DT2<-exp(mvrnorm(n2,mu=mu2,Sigma=Sigma2))
z <- rep(c(1,0), c(n1,n2))
DT<-data.frame(rbind(DT1,DT2))
data2<-cbind(DT,z)
data<-data2
# apply model
dataa<-xgb.DMatrix(data.matrix(data[, -which(names(data) %in% c("z"))]),
label=data$z)
predictions <- predict(model, dataa)
predictions <- ifelse(predictions<cutoff, 0, 1)
p<-prediction(predictions,data[,dim(data)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
youden_xg<-c(youden_xg,maxx)
count<-count+1

}

config<-c(config,configuracion[ind])
mean_youden_xg<-c(mean_youden_xg, mean(youden_xg))
sd_youden_xg<-c(sd_youden_xg, sd(youden_xg))

}

# Final results
tabla <- data.frame(config, mean_youden_xg,sd_youden_xg)

```

B.9. Ten biomarkers. Log-normal distributions. Min-max approach

```

# Authors: Rocio Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean and standard deviation of the maximum Youden
indices obtained through min-max approach on various lognormal simulated data
scenarios of ten biomarkers.
# Imports needed: MASS, SLModels
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
Youden Index', and reference the SLModels library.

library(MASS)
library(SLModels)

mean_youden_mm<-c()
sd_youden_mm<-c()
config<-c()

#Run configuration code for random sample generation#

for (ind in seq(1,length(mu11),1)){

  mu1<-mu11[ind][[1]]
  mu2<-mu22[ind][[1]]
  Sigma1<-Sigma11[ind][[1]]

```

```

Sigma2<-Sigma22[ind][[1]]
youden_mm<-c()

sequence <- seq(1,100,1)
sequence2<- seq(101,201,1)
n1<-50 #500
n2<-50 #500
count<-1

for(x in sequence){

  ## Training ##
  set.seed(x)
  # create dataset
  DT1<-exp(mvrnorm(n1,mu=mu1,Sigma=Sigma1))
  DT2<-exp(mvrnorm(n2,mu=mu2,Sigma=Sigma2))
  z <- rep(c(1,0), c(n1,n2))
  DT<-data.frame(rbind(DT1,DT2))
  data2<-cbind(DT,z)
  data<-data2
  # train model
  a<-SLModels(data,algorithm="minmax")

  ## Validation ##
  set.seed(sequence2[count])
  # create dataset
  DT1<-mvrnorm(n1,mu=mu1,Sigma=Sigma1)
  DT2<-mvrnorm(n2,mu=mu2,Sigma=Sigma2)
  z <- rep(c(1,0), c(n1,n2))
  DT<-data.frame(rbind(DT1,DT2))
  data2<-cbind(DT,z)
  data<-data2
  # apply model
  aa<-cbind(apply(data[, c(1,2,3,4,5,6,7,8,9,10)],1,max),apply(data[, c
    (1,2,3,4,5,6,7,8,9,10)],1,min))
  data22<-matrix(unlist(a['max'])*unlist(aa[,1])+unlist(a['min'])*unlist(aa
    [,2]))
  data222<-data22[,1]
  data222 <- ifelse(data222<unlist(a['Cutoff']), 0, 1)
  p<-prediction(data222,data[,dim(data)[2]])
  at<-attributes(performance(p,"sens","spec"))
  maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
  youden_mm<-c(youden_mm,maxx)
  count<-count+1

}

config<-c(config,configuracion[ind])
mean_youden_mm<-c(mean_youden_mm, mean(youden_mm))
sd_youden_mm<-c(sd_youden_mm, sd(youden_mm))

}

# Final results
tabla <- data.frame(config, mean_youden_mm,sd_youden_mm)

```

B.10. Ten biomarkers. Log-normal distributions. Min-Max-Median approach

```

# Authors: Rocio Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean and standard deviation of the maximum Youden
  indices obtained through min-max-median approach on various lognormal simulated
  data scenarios of ten biomarkers.
# Imports needed: MASS, SLModels
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
  IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
  Youden Index', and reference the SLModels library.

library(MASS)
library(SLModels)

mean_youden_mmm<-c()

```

```

sd_youden_mmm<-c()
config<-c()

#Run configuration code for random sample generation#

for (ind in seq(1,length(mu11),1)){

  mu1<-mu11[ind][[1]]
  mu2<-mu22[ind][[1]]
  Sigma1<-Sigma11[ind][[1]]
  Sigma2<-Sigma22[ind][[1]]
  youden_mmm<-c()

  sequence <- seq(1,100,1)
  sequence2<- seq(101,201,1)
  n1<-50 #500
  n2<-50 #500
  count<-1

  for(x in sequence){

    ## Training ##
    set.seed(x)
    # create dataset
    DT1<-exp(mvrnorm(n1,mu=mu1,Sigma=Sigma1))
    DT2<-exp(mvrnorm(n2,mu=mu2,Sigma=Sigma2))
    z <- rep(c(1,0), c(n1,n2))
    DT<-data.frame(rbind(DT1,DT2))
    data2<-cbind(DT,z)
    data<-data2
    # train model
    a<-SLModels(data,algorithm="minmaxmedian")

    ## Validation ##
    set.seed(sequence2[count])
    # create dataset
    DT1<-exp(mvrnorm(n1,mu=mu1,Sigma=Sigma1))
    DT2<-exp(mvrnorm(n2,mu=mu2,Sigma=Sigma2))
    z <- rep(c(1,0), c(n1,n2))
    DT<-data.frame(rbind(DT1,DT2))
    data2<-cbind(DT,z)
    data<-data2
    # apply model
    aa<-cbind(apply(data[,c(1,2,3,4,5,6,7,8,9,10)],1,max),apply(data[,c
      (1,2,3,4,5,6,7,8,9,10)],1,min),apply(data[,c(1,2,3,4,5,6,7,8,9,10)],1,
      median))
    data22<-matrix(unlist(a['max'])*unlist(aa[,1])+unlist(a['min'])*unlist(aa
      [,2])+unlist(a['median'])*unlist(aa[,3]))
    data22<-data22[,1]
    data222 <- ifelse(data22<unlist(a['Cutoff']), 0, 1)
    p<-prediction(data22,data[,dim(data)[2]])
    at<-attributes(performance(p,"sens","spec"))
    maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
    youden_mmm<-c(youden_mmm,maxx)
    count<-count+1

  }

  config<-c(config,configuracion[ind])
  mean_youden_mmm<-c(mean_youden_mmm, mean(youden_mmm))
  sd_youden_mmm<-c(sd_youden_mmm, sd(youden_mmm))

}

# Final results
tabla <- data.frame(config, mean_youden_mmm,sd_youden_mmm)

```

B.11. Ten biomarkers. Log-normal distributions. Min-Max-IQR approach

```
# Authors: Rocio Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
```

```

# Description: This code calculates the mean and standard deviation of the maximum Youden
indices obtained through min-max-IQR approach on various lognormal simulated data
scenarios of ten biomarkers.
# Imports needed: MASS, SLModels
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
Youden Index', and reference the SLModels library.

library(MASS)
library(SLModels)

mean_youden_mmiqr<-c()
sd_youden_mmiqr<-c()
config<-c()

#Run configuration code for random sample generation#

for (ind in seq(1,length(mu11),1)){

  mu1<-mu11[ind][[1]]
  mu2<-mu22[ind][[1]]
  Sigma1<-Sigma11[ind][[1]]
  Sigma2<-Sigma22[ind][[1]]
  youden_mmiqr<-c()

  sequence <- seq(1,100,1)
  sequence2<- seq(101,201,1)
  n1<-50 #500
  n2<-50 #500
  count<-1

  for(x in sequence){

    ## Training ##
    set.seed(x)
    # create dataset
    DT1<-exp(mvrnorm(n1,mu=mu1,Sigma=Sigma1))
    DT2<-exp(mvrnorm(n2,mu=mu2,Sigma=Sigma2))
    z <- rep(c(1,0), c(n1,n2))
    DT<-data.frame(rbind(DT1,DT2))
    data2<-cbind(DT,z)
    data<-data2
    # train model
    a<-SLModels(data,algorithm="minmaxiqr")

    ## Validation ##
    set.seed(sequence2[count])
    # create dataset
    DT1<-exp(mvrnorm(n1,mu=mu1,Sigma=Sigma1))
    DT2<-exp(mvrnorm(n2,mu=mu2,Sigma=Sigma2))
    z <- rep(c(1,0), c(n1,n2))
    DT<-data.frame(rbind(DT1,DT2))
    data2<-cbind(DT,z)
    data<-data2
    # apply model
    aa<-cbind(apply(data[,c(1,2,3,4,5,6,7,8,9,10)],1,max),apply(data[,c
      (1,2,3,4,5,6,7,8,9,10)],1,min),apply(data[,c(1,2,3,4,5,6,7,8,9,10)],1,
      quantile)[3,]-apply(data[,c(1,2,3,4,5,6,7,8,9,10)],1,quantile)[1,])
    data22<-matrix(unlist(a['max'])*unlist(aa[,1])+unlist(a['min'])*unlist(aa
      [,2])+unlist(a['iqr'])*unlist(aa[,3]))
    data222<-data22[,1]
    data222 <- ifelse(data222<unlist(a['Cutoff']), 0, 1)
    p<-prediction(data222,data[,dim(data)[2]])
    at<-attributes(performance(p,"sens","spec"))
    maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
    youden_mmiqr<-c(youden_mmiqr,maxx)
    count<-count+1

  }

  config<-c(config,configuracion[ind])
  mean_youden_mmiqr<-c(mean_youden_mmiqr, mean(youden_mmiqr))

```

```

sd_youden_mmiqr<-c(sd_youden_mmiqr, sd(youden_mmiqr))
}

# Final results
tabla <- data.frame(config, mean_youden_mmiqr,sd_youden_mmiqr)

```

2. REAL DATASETS

A. DMD dataset

A.1. Descriptive analysis

```

# Authors: Rocio Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code analyses the information of the DMD (Duchenne Muscular Dystrophy
) dataset. Specifically, it plots the distribution of biomarkers between the carrier
and non-carrier groups, calculates the correlations between biomarkers and the
univariate estimated Youden index.
# Dataset: The Duchenne muscular dystrophy dataset can be found at https://hbiostat.org/
data/
# Imports needed: ROCR
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
Youden Index'.

library(ROCR)

set.seed(2)

# Load dataset
datos <- read.csv("dmd.csv", sep=",")
colSums(is.na(datos))
datos<-na.omit(datos, cols=c("pk", "ld"))
data<-datos[,c(5,6,7,8,9)]
colnames(data)<-c('ck','h','pk','ld','Carrier')
data<-data[
  with(data, order(Carrier)),
]
n1<-dim(data[data$Carrier == 0,])[1]
n2<-dim(data[data$Carrier == 1,])[1]

# Distributions
par(mfrow=c(2,4))
boxplot(ck ~ Carrier, data = data,col = c("#FFE0B2", "#FFA726"),names = c("NO","YES"))
boxplot(h ~ Carrier, data = data,col = c("chartreuse3", "chartreuse4"),names = c("NO",
YES"))
boxplot(pk ~ Carrier, data = data,col = c("brown3", "brown4"),names = c("NO","YES"))
boxplot(ld ~ Carrier, data = data,col = c("aquamarine3", "aquamarine4"),names = c("NO",
YES"))

# Correlations
cor(data[which(data$Carrier==0),-c(5)]) #non-carrier
cor(data[which(data$Carrier==1),-c(5)]) #carrier

# Univariate. Youden index
for(i in seq(1,4,1)){
  print(i)
  p<-prediction(as.numeric(data[,i]),data[,dim(data)[2]])
  at<-attributes(performance(p,"sens","spec"))
  maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
  print(paste('youden',round(maxx,3)))
}

```

A.2. Performance results

```

# Authors: Rocio Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean of the maximum Youden indices, as well as
the sensitivity and specificity, obtained after applying the logistic regression,
xgboost algorithm, min-max approach, min-max-median approach and min-max-IQR
approach, on the DMD (Duchenne Muscular Dystrophy) dataset.

```



```

# Dataset: The Duchenne muscular dystrophy dataset can be found at https://hbiostat.org/
data/
# Imports needed: MASS, SLModels, caret, xgboost
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
Youden Index', and reference the SLModels library.

library(MASS)
library(SLModels)
library(caret)
library(xgboost)

set.seed(2)

# Load dataset
datos <- read.csv("dmd.csv", sep=",")
colSums(is.na(datos))
datos<-na.omit(datos, cols=c("pk", "ld"))
data<-datos[,c(5,6,7,8,9)]
datos<-data
colnames(datos)<-c("V1", "V2", "V3", "V4", "z")
data<-datos
data<-data[
  with(data, order(z)),
]
n1<-dim(data[data$z == 0,])[1]
n2<-dim(data[data$z == 1,])[1]

data1<-data
folds <- createFolds(data1$z, k = 10)
data1 <- data.frame(data1$V1, data1$V2, data1$V3, data1$V4, data1$z)

data10<-data1
media1<-mean(data10[,1])
media2<-mean(data10[,2])
media3<-mean(data10[,3])
media4<-mean(data10[,4])
sd1<-sd(data10[,1])
sd2<-sd(data10[,2])
sd3<-sd(data10[,3])
sd4<-sd(data10[,4])

for (row in 1:dim(data1)[1]) {
  data10[row, 1] <- (data10[row, 1]-media1)/sd1
  data10[row, 2] <- (data10[row, 2]-media2)/sd2
  data10[row, 3] <- (data10[row, 3]-media3)/sd3
  data10[row, 4] <- (data10[row, 4]-media4)/sd4
}

youdenSummary <- function(data, lev = NULL, model = NULL){
  if (length(lev) > 2) {
    stop(paste("Your outcome has", length(lev), "levels. The youdenSummary() function isn't
appropriate."))
  }
  if (!all(levels(data[, "pred"]) == lev)) {
    stop("levels of observed and predicted data do not match")
  }
  Sens <- caret::sensitivity(data[, "pred"], data[, "obs"], lev[1])
  Spec <- caret::specificity(data[, "pred"], data[, "obs"], lev[2])
  j <- Sens + Spec
  out <- c(j, Spec, Sens)
  names(out) <- c("j", "Spec", "Sens")
  out
}

youden_log <-c()
sensitivity_log<-c()
specificity_log<-c()
youden_xg <-c()
sensitivity_xg<-c()
specificity_xg<-c()

```

```

youden_mm <-c()
sensitivity_mm<-c()
specificity_mm<-c()
youden_mmm <-c()
sensitivity_mmm<-c()
specificity_mmm<-c()
auc_mmiqr<-c()
acc_mmiqr<-c()
youden_mmiqr <-c()
sensitivity_mmiqr<-c()
specificity_mmiqr<-c()

# 10-fold cross validation
for (ff in folds){

  # Split training and validation
  train <- data1[-ff,]
  valid <- data1[ff,]
  train<-train[,c(1,2,3,4,5)]
  valid<-valid[,c(1,2,3,4,5)]
  train0 <- data10[-ff,]
  valid0 <- data10[ff,]
  train0<-train0[,c(1,2,3,4,5)]
  valid0<-valid0[,c(1,2,3,4,5)]

  ## Training ##
  # Logistic
  logistic<-glm(data1.z ~ ., train,family="binomial")
  aa<-unlist(logistic$coefficients[2])*unlist(train[,1])+unlist(logistic$
    coefficients[3])*unlist(train[,2])+unlist(logistic$coefficients[4])*unlist(
    train[,3])+unlist(logistic$coefficients[5])*unlist(train[,4])
  data2<-matrix(aa)
  p<-prediction(data2,train[,dim(train)[2]])
  a<-attributes(performance(p,"sens","spec"))
  cutoffflog<-a$alpha.values[[1]][which.max(a$y.values[[1]]+a$x.values[[1]]-1)]
  # XGBoost
  train[, which(names(train) %in% c("data1.z"))]<-as.factor(train[, which(names(
    train) %in% c("data1.z"))])
  train_control = trainControl(method = "cv", number = 5, search = "grid",
    summaryFunction = youdenSummary)
  gbmGrid <- expand.grid(max_depth = c(2,3,4,5,6,10), nrounds = c(50,100,200), eta =
    c(0.1, 0.3), gamma = c(0, 0.5), subsample = c(0.5, 1.0), min_child_weight =
    1, colsample_bytree = c(0.5,1))
  model0 = train(data1.z~., data = train, method = "xgbTree", metric="j", trControl
    = train_control, tuneGrid = gbmGrid)
  hipers <- model0$finalModel$tuneValue
  train$data1.z<-as.numeric(as.character(train$data1.z))
  model <- xgboost(data = data.matrix(train[, -which(names(train) %in% c("data1.z")
    ))), label=as.numeric(train$data1.z), objective = "binary:logistic", early_
    stopping_rounds = 10,nthread = 2, max.depth = hipers$max_depth, eta = hipers$
    eta, nrounds = hipers$nrounds, gamma = hipers$gamma, colsample_bytree =
    hipers$colsample_bytree, min_child_weight=hipers$min_child_weight, subsample=
    hipers$subsample)
  dataaa<-xgb.DMatrix(data.matrix(train[, -which(names(train) %in% c("data1.z"))]),
    label=train$data1.z)
  predictions <- predict(model, dataaa)
  p<-prediction(predictions,train[,dim(train)[2]])
  a<-attributes(performance(p,"sens","spec"))
  cutofffxg<-a$alpha.values[[1]][which.max(a$y.values[[1]]+a$x.values[[1]]-1)]
  # MM
  a_mm<-SLModels(train0,algorithm="minmax")
  # MMM
  a_mmm<-SLModels(train0,algorithm="minmaxmedian")
  # MMIQR
  a_mmiqr<-SLModels(train0,algorithm="minmaxiqr")

  ## Validation ##
  # Logistic
  aa<-unlist(logistic$coefficients[2])*unlist(valid[,1])+unlist(logistic$
    coefficients[3])*unlist(valid[,2])+unlist(logistic$coefficients[4])*unlist(
    valid[,3])+unlist(logistic$coefficients[5])*unlist(valid[,4])
  data22<-matrix(aa)

```

```

data222<-data22[,1]
data222 <- ifelse(data222<cutofflog, 0, 1)
p<-prediction(data222,valid[,dim(valid)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
youden_log<-c(youden_log,maxx)
sensitivity_log<-c(sensitivity_log,at$y.values[[1]][which.max(at$y.values[[1]]+at$x.
values[[1]]-1)])
specificity_log<-c(specificity_log,at$x.values[[1]][which.max(at$y.values[[1]]+at$x.
values[[1]]-1)])
# XGBoost
valid[, which(names(valid) %in% c("data1.z"))]<-as.factor(valid[, which(names(
valid) %in% c("data1.z"))])
valid$data1.z<-as.numeric(as.character(valid$data1.z))
dataa<-xgb.DMatrix(data.matrix(valid[, -which(names(valid) %in% c("data1.z"))]),
label=valid$data1.z)
predictions <- predict(model, dataa)
predictions <- ifelse(predictions<cutoffxg, 0, 1)
p<-prediction(predictions,valid[,dim(valid)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
youden_xg<-c(youden_xg,maxx)
sensitivity_xg<-c(sensitivity_xg,at$y.values[[1]][which.max(at$y.values[[1]]+at$x.
values[[1]]-1)])
specificity_xg<-c(specificity_xg,at$x.values[[1]][which.max(at$y.values[[1]]+at$x.
values[[1]]-1)])
# MM
aa<-cbind(apply(valid0[,c(1,2,3,4)],1,max),apply(valid0[,c(1,2,3,4)],1,min))
data22<-matrix(unlist(a_mm['max'])*unlist(aa[,1])+unlist(a_mm['min'])*unlist(aa
[,2]))
data222<-data22[,1]
data222 <- ifelse(data222<unlist(a_mm['Cutoff']), 0, 1)
p<-prediction(data222,valid0[,dim(valid0)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
youden_mm<-c(youden_mm,maxx)
sensitivity_mm<-c(sensitivity_mm,at$y.values[[1]][which.max(at$y.values[[1]]+at$x.
values[[1]]-1)])
specificity_mm<-c(specificity_mm,at$x.values[[1]][which.max(at$y.values[[1]]+at$x.
values[[1]]-1)])
# MMM
aa<-cbind(apply(valid0[,c(1,2,3,4)],1,max),apply(valid0[,c(1,2,3,4)],1,min),apply(
valid0[,c(1,2,3,4)],1,median))
data22<-matrix(unlist(a_mmm['max'])*unlist(aa[,1])+unlist(a_mmm['min'])*unlist(aa
[,2])+unlist(a_mmm['median'])*unlist(aa[,3]))
data222<-data22[,1]
data222 <- ifelse(data222<unlist(a_mmm['Cutoff']), 0, 1)
p<-prediction(data222,valid0[,dim(valid0)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
youden_mmm<-c(youden_mmm,maxx)
sensitivity_mmm<-c(sensitivity_mmm,at$y.values[[1]][which.max(at$y.values[[1]]+at$x.
values[[1]]-1)])
specificity_mmm<-c(specificity_mmm,at$x.values[[1]][which.max(at$y.values[[1]]+at$x.
values[[1]]-1)])
# MMIQR
aa<-cbind(apply(valid0[,c(1,2,3,4)],1,max),apply(valid0[,c(1,2,3,4)],1,min),apply(
valid0[,c(1,2,3,4)],1,quantile)[3,]-apply(valid0[,c(1,2,3,4)],1,quantile)
[1,])
data22<-matrix(unlist(a_mmiqr['max'])*unlist(aa[,1])+unlist(a_mmiqr['min'])*unlist
(aa[,2])+unlist(a_mmiqr['iqr'])*unlist(aa[,3]))
data222<-data22[,1]
data222 <- ifelse(data222<unlist(a_mmiqr['Cutoff']), 0, 1)
p<-prediction(data222,valid0[,dim(valid0)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
youden_mmiqr<-c(youden_mmiqr,maxx)
sensitivity_mmiqr<-c(sensitivity_mmiqr,at$y.values[[1]][which.max(at$y.values[[1]]+at
$x.values[[1]]-1)])
specificity_mmiqr<-c(specificity_mmiqr,at$x.values[[1]][which.max(at$y.values[[1]]+at
$x.values[[1]]-1)])

```

```

}

# Final results
config<-c("logistic", "xgboost", "mm", "mmm", "mmiqr")
mean_youden_log<-c(mean(youden_log), mean(youden_xg), mean(youden_mm), mean(youden_mmm),
  mean(youden_mmiqr))
mean_sensitivity_log<-c(mean(sensitivity_log), mean(sensitivity_xg), mean(sensitivity_mm),
  mean(sensitivity_mmm), mean(sensitivity_mmiqr))
mean_specificity_log<-c(mean(specificity_log), mean(specificity_xg), mean(specificity_mm),
  mean(specificity_mmm), mean(specificity_mmiqr))
tabla <- data.frame(config, mean_youden_log, mean_sensitivity_log, mean_specificity_log)

```

B. Maternal Health Risk dataset. High-Medium vs. Low Risk

B.1. Descriptive analysis

```

# Authors: Rocio Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code analyses the information of the Maternal Health Risk dataset (
  high-medium vs. low risk). Specifically, it plots the distribution of biomarkers
  between the carrier and non-carrier groups, calculates the correlations between
  biomarkers and the univariate estimated Youden index.
# Dataset: The Maternal Health Risk dataset can be found at http://archive.ics.uci.edu/ml
# Imports needed: ROCR
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
  IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
  Youden Index'.

library(ROCR)

set.seed(2)

# Load dataset
datos <- read.csv("maternal.csv", sep=",", header=TRUE)
datos$RiskLevel<-unclass(datos$RiskLevel)
datos$RiskLevel <- replace(datos$RiskLevel, datos$RiskLevel=="low risk", 0)
datos$RiskLevel <- replace(datos$RiskLevel, datos$RiskLevel!=0, 1)
datos<-datos[which(datos$Age>=13 & datos$Age<=50 & datos$HeartRate>7),]
datos<-datos[!duplicated(datos),]
data<-datos
data<-data[
  with(data, order(RiskLevel)),
]
n1<-dim(data[data$RiskLevel == 0,])[1]
n2<-dim(data[data$RiskLevel == 1,])[1]

# Distributions
par(mfrow=c(2,4))
boxplot(Age ~ RiskLevel, data = data, names = c("Low", ">=Med"), col = c("#FFE0B2", "#FFA726"),
  outpch = 24)
boxplot(SystolicBP ~ RiskLevel, data = data, names = c("Low", ">=Med"), col = c("chartreuse3", "chartreuse4"),
  outpch = 24)
boxplot(DiastolicBP ~ RiskLevel, data = data, names = c("Low", ">=Med"), col = c("brown3", "brown4"),
  outpch = 24)
boxplot(BS ~ RiskLevel, data = data, names = c("Low", ">=Med"), col = c("aquamarine3", "aquamarine4"),
  outpch = 24)
boxplot(BodyTemp ~ RiskLevel, data = data, names = c("Low", ">=Med"), col = c("azure4", "azure4"),
  outpch = 24)
boxplot(HeartRate ~ RiskLevel, data = data, names = c("Low", ">=Med"), col = c("darkorchid3", "darkorchid4"),
  outpch = 24)

# Correlations
cor(data[which(data$RiskLevel==0),-c(7)]) #low risk
cor(data[which(data$RiskLevel==1),-c(7)]) #high-medium risk

# Univariate. Youden index
for(i in seq(1,6,1)){
  print(i)
  p<-prediction(as.numeric(data[,i]), data[,dim(data)[2]])
  at<-attributes(performance(p, "sens", "spec"))
  maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)

```

```

    print(paste('youden',round(maxx,3)))
}

```

B.2. Performance results

```

# Authors: Rocio Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean of the maximum Youden indices, as well as
the sensitivity and specificity, obtained after applying the logistic regression,
xgboost algorithm, min-max approach, min-max-median approach and min-max-IQR
approach, on the Maternal Health Risk dataset (high-medium vs. low risk).
# Dataset: The Maternal Health Risk dataset can be found at http://archive.ics.uci.edu/ml
# Imports needed: MASS, SLModels, caret, xgboost
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
Youden Index', and reference the SLModels library.

library(MASS)
library(SLModels)
library(caret)
library(xgboost)

set.seed(2)

# Load dataset
datos <- read.csv("maternal.csv", sep=",", header=TRUE)
datos$RiskLevel<-unclass(datos$RiskLevel)
datos$RiskLevel <- replace(datos$RiskLevel, datos$RiskLevel=="low risk", 0)
datos$RiskLevel <- replace(datos$RiskLevel, datos$RiskLevel!=0, 1)
datos<-datos[which(datos$Age>=13 & datos$Age<=50 & datos$HeartRate>7),]
datos<-datos[!duplicated(datos),]
colnames(datos)<-c("V1", "V2", "V3", "V4", "V5", "V6", "z")
data<-datos
data<-data[
  with(data, order(z)),
]
n1<-dim(data[data$z == 0,])[1]
n2<-dim(data[data$z == 1,])[1]
data$z <- as.numeric(data$z)

data1<-data
folds <- createFolds(data1$z, k = 10)
data1 <- data.frame(data1$V1, data1$V2, data1$V3, data1$V4, data1$V5,data1$V6, data1$z)

data10<-data1
media1<-mean(data10[,1])
media2<-mean(data10[,2])
media3<-mean(data10[,3])
media4<-mean(data10[,4])
media5<-mean(data10[,5])
media6<-mean(data10[,6])
sd1<-sd(data10[,1])
sd2<-sd(data10[,2])
sd3<-sd(data10[,3])
sd4<-sd(data10[,4])
sd5<-sd(data10[,5])
sd6<-sd(data10[,6])

for (row in 1:dim(data1)[1]) {
  data10[row, 1] <- (data10[row, 1]-media1)/sd1
  data10[row, 2] <- (data10[row, 2]-media2)/sd2
  data10[row, 3] <- (data10[row, 3]-media3)/sd3
  data10[row, 4] <- (data10[row, 4]-media4)/sd4
  data10[row, 5] <- (data10[row, 5]-media5)/sd5
  data10[row, 6] <- (data10[row, 6]-media6)/sd6
}

youdenSummary <- function(data, lev = NULL, model = NULL){
  if (length(lev) > 2) {
    stop(paste("Your outcome has", length(lev), "levels. The youdenSummary() function isn't
appropriate."))
  }
}

```

```

if (!all(levels(data[, "pred"]) == lev)) {
  stop("levels of observed and predicted data do not match")
}
Sens <- caret::sensitivity(data[, "pred"], data[, "obs"], lev[1])
Spec <- caret::specificity(data[, "pred"], data[, "obs"], lev[2])
j <- Sens + Spec
out <- c(j, Spec, Sens)
names(out) <- c("j", "Spec", "Sens")
out
}

youden_log <-c()
sensitivity_log<-c()
specificity_log<-c()
youden_xg <-c()
sensitivity_xg<-c()
specificity_xg<-c()
youden_mm <-c()
sensitivity_mm<-c()
specificity_mm<-c()
youden_mmm <-c()
sensitivity_mmm<-c()
specificity_mmm<-c()
auc_mmiqr<-c()
acc_mmiqr<-c()
youden_mmiqr <-c()
sensitivity_mmiqr<-c()
specificity_mmiqr<-c()

# 10-fold cross validation
for (ff in folds){

  # Split training and validation
  train <- data1[-ff,]
  valid <- data1[ff,]
  train<-train[,c(1,2,3,4,5,6,7)]
  valid<-valid[,c(1,2,3,4,5,6,7)]
  train0 <- data10[-ff,]
  valid0 <- data10[ff,]
  train0<-train0[,c(1,2,3,4,5,6,7)]
  valid0<-valid0[,c(1,2,3,4,5,6,7)]

  ## Training ##
  # Logistic
  logistic<-glm(data1.z ~ ., train,family="binomial")
  aa<-unlist(logistic$coefficients[2])*unlist(train[,1])+unlist(logistic$
    coefficients[3])*unlist(train[,2])+unlist(logistic$coefficients[4])*unlist(
    train[,3])+unlist(logistic$coefficients[5])*unlist(train[,4])+unlist(logistic
    $coefficients[6])*unlist(train[,5])+unlist(logistic$coefficients[7])*unlist(
    train[,6])
  data2<-matrix(aa)
  p<-prediction(data2,train[,dim(train)[2]])
  a<-attributes(performance(p,"sens","spec"))
  cutofflog<-a$alpha.values[[1]][which.max(a$y.values[[1]]+a$x.values[[1]]-1)]
  # XGBoost
  train[, which(names(train) %in% c("data1.z"))]<-as.factor(train[, which(names(
    train) %in% c("data1.z"))])
  train_control = trainControl(method = "cv", number = 5, search = "grid",
    summaryFunction = youdenSumary)
  gbmGrid <- expand.grid(max_depth = c(2,3,6,8,10,20), nrounds = c(50,100,200), eta
    = c(0.1, 0.3), gamma = c(0, 0.5), subsample = c(0.5, 1.0), min_child_weight =
    1, colsample_bytree = c(0.5,1))
  model0 <- train(data1.z~., data = train, method = "xgbTree", metric="j", trControl
    = train_control, tuneGrid = gbmGrid)
  hipers <- model0$finalModel$tuneValue
  train$data1.z<-as.numeric(as.character(train$data1.z))
  model <- xgboost(data = data.matrix(train[, -which(names(train) %in% c("data1.z")
    )]), label=as.numeric(train$data1.z), objective = "binary:logistic", early_
    stopping_rounds = 10,nthread = 2, max.depth = hipers$max_depth, eta = hipers$
    eta, nrounds = hipers$nrounds, gamma = hipers$gamma, colsample_bytree =
    hipers$colsample_bytree, min_child_weight=hipers$min_child_weight, subsample=
    hipers$subsample)

```

```

dataaa<-xgb.DMatrix(data.matrix(train[, ~which(names(train) %in% c("data1.z"))]),
  label=train$data1.z)
predictions <- predict(model, dataaa)
p<-prediction(predictions,train[,dim(train)[2]])
a<-attributes(performance(p,"sens","spec"))
cutoffxg<-a$alpha.values[[1]][which.max(a$y.values[[1]]+a$x.values[[1]]-1)]
# MM
a_mm<-SLModels(train0,algorithm="minmax")
# MMM
a_mmm<-SLModels(train0,algorithm="minmaxmedian")
# MMIQR
a_mmiqr<-SLModels(train0,algorithm="minmaxiqr")

## Validation ##
# Logistic
aa<-unlist(logistic$coefficients[2])*unlist(valid[,1])+unlist(logistic$
  coefficients[3])*unlist(valid[,2])+unlist(logistic$coefficients[4])*unlist(
  valid[,3])+unlist(logistic$coefficients[5])*unlist(valid[,4])+unlist(logistic
  $coefficients[6])*unlist(valid[,5])+unlist(logistic$coefficients[7])*unlist(
  valid[,6])
data22<-matrix(aa)
data222<-data22[,1]
data222 <- ifelse(data222<cutofflog, 0, 1)
p<-prediction(data222,valid[,dim(valid)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
youden_log<-c(youden_log,maxx)
sensitivity_log<-c(sensitivity_log,at$y.values[[1]][which.max(at$y.values[[1]]+at$x.
  values[[1]]-1)])
specificity_log<-c(specificity_log,at$x.values[[1]][which.max(at$y.values[[1]]+at$x.
  values[[1]]-1)])
# XGBoost
valid[, which(names(valid) %in% c("data1.z"))]<-as.factor(valid[, which(names(
  valid) %in% c("data1.z"))])
valid$data1.z<-as.numeric(as.character(valid$data1.z))
dataaa<-xgb.DMatrix(data.matrix(valid[, ~which(names(valid) %in% c("data1.z"))]),
  label=valid$data1.z)
predictions <- predict(model, dataaa)
predictions <- ifelse(predictions<cutoffxg, 0, 1)
p<-prediction(predictions,valid[,dim(valid)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
youden_xg<-c(youden_xg,maxx)
sensitivity_xg<-c(sensitivity_xg,at$y.values[[1]][which.max(at$y.values[[1]]+at$x.
  values[[1]]-1)])
specificity_xg<-c(specificity_xg,at$x.values[[1]][which.max(at$y.values[[1]]+at$x.
  values[[1]]-1)])
# MM
aa<-cbind(apply(valid0[,c(1,2,3,4,5,6)],1,max),apply(valid0[,c(1,2,3,4,5,6)],1,min
))
data22<-matrix(unlist(a_mm['max'])*unlist(aa[,1])+unlist(a_mm['min'])*unlist(aa
[,2]))
data222<-data22[,1]
data222 <- ifelse(data222<unlist(a_mm['Cutoff']), 0, 1)
p<-prediction(data222,valid0[,dim(valid0)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
youden_mm<-c(youden_mm,maxx)
sensitivity_mm<-c(sensitivity_mm,at$y.values[[1]][which.max(at$y.values[[1]]+at$x.
  values[[1]]-1)])
specificity_mm<-c(specificity_mm,at$x.values[[1]][which.max(at$y.values[[1]]+at$x.
  values[[1]]-1)])
# MMM
aa<-cbind(apply(valid0[,c(1,2,3,4,5,6)],1,max),apply(valid0[,c(1,2,3,4,5,6)],1,min
),apply(valid0[,c(1,2,3,4,5,6)],1,median))
data22<-matrix(unlist(a_mmm['max'])*unlist(aa[,1])+unlist(a_mmm['min'])*unlist(aa
[,2])+unlist(a_mmm['median'])*unlist(aa[,3]))
data222<-data22[,1]
data222 <- ifelse(data222<unlist(a_mmm['Cutoff']), 0, 1)
p<-prediction(data222,valid0[,dim(valid0)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)

```

```

        youden_mmm<-c(youden_mmm,maxx)
        sensitivity_mmm<-c(sensitivity_mmm,at$y.values[[1]][which.max(at$y.values[[1]]+at$x.
            values[[1]]-1)])
        specificity_mmm<-c(specificity_mmm,at$x.values[[1]][which.max(at$y.values[[1]]+at$x.
            values[[1]]-1)])
        # MMIQR
        aa<-cbind(apply(valid0[,c(1,2,3,4,5,6)],1,max),apply(valid0[,c(1,2,3,4,5,6)],1,min
            ),apply(valid0[,c(1,2,3,4,5,6)],1,quantile)[3,]-apply(valid0[,c(1,2,3,4,5,6)
            ],1,quantile)[1,])
        data22<-matrix(unlist(a_mmiqr['max'])*unlist(aa[,1])+unlist(a_mmiqr['min'])*unlist
            (aa[,2])+unlist(a_mmiqr['iqr'])*unlist(aa[,3]))
        data222<-data22[,1]
        data222 <- ifelse(data22<unlist(a_mmiqr['Cutoff']), 0, 1)
        p<-prediction(data222,valid0[,dim(valid0)[2]])
        at<-attributes(performance(p,"sens","spec"))
        maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
        youden_mmiqr<-c(youden_mmiqr,maxx)
        sensitivity_mmiqr<-c(sensitivity_mmiqr,at$y.values[[1]][which.max(at$y.values[[1]]+at
            $x.values[[1]]-1)])
        specificity_mmiqr<-c(specificity_mmiqr,at$x.values[[1]][which.max(at$y.values[[1]]+at
            $x.values[[1]]-1)])
    }

    # Final results
    config<-c("logistic", "xgboost","mm", "mmm", "mmiqr")
    mean_youden_log<-c(mean(youden_log),mean(youden_xg), mean(youden_mm), mean(youden_mmm),
        mean(youden_mmiqr))
    mean_sensitivity_log<-c(mean(sensitivity_log),mean(sensitivity_xg), mean(sensitivity_mm),
        mean(sensitivity_mmm), mean(sensitivity_mmiqr))
    mean_specificity_log<-c(mean(specificity_log),mean(specificity_xg), mean(specificity_mm),
        mean(specificity_mmm), mean(specificity_mmiqr))
    tabla <- data.frame(config, mean_youden_log,mean_sensitivity_log,mean_specificity_log)

```

C. Maternal Health Risk dataset. High vs. Medium-Low Risk

C.1. Descriptive analysis

```

# Authors: Rocio Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code analyses the information of the Maternal Health Risk dataset (
    high vs. medium-low risk). Specifically, it plots the distribution of biomarkers
    between the carrier and non-carrier groups, calculates the correlations between
    biomarkers and the univariate estimated Youden index.
# Dataset: The Maternal Health Risk dataset can be found at http://archive.ics.uci.edu/ml
# Imports needed: ROCR
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
    IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
    Youden Index'.

library(ROCR)

set.seed(2)

# Load dataset
datos <- read.csv("maternal.csv", sep=",", header=TRUE)
datos$RiskLevel<-unclass(datos$RiskLevel)
datos$RiskLevel <- replace(datos$RiskLevel, datos$RiskLevel=="high risk", 1)
datos$RiskLevel <- replace(datos$RiskLevel, datos$RiskLevel!=1, 0)
datos<-datos[which(datos$Age>=13 & datos$Age<=50 & datos$HeartRate>7),]
datos<-datos[!duplicated(datos),]
data<-datos
data<-data[
    with(data, order(RiskLevel)),
]
n1<-dim(data[data$RiskLevel == 0,])[1]
n2<-dim(data[data$RiskLevel == 1,])[1]

# Distributions
par(mfrow=c(2,4))
boxplot(Age ~ RiskLevel, data = data,names = c("<=Med", "High"),col = c("#FFD700", "#
    FFA726"),outpch = 24)

```



```

boxplot(SystolicBP ~ RiskLevel, data = data, names = c("<=Med", "High"), col = c("
  chartreuse3", "chartreuse4"), outpch = 24)
boxplot(DiastolicBP ~ RiskLevel, data = data, names = c("<=Med", "High"), col = c("brown3",
  "brown4"), outpch = 24)
boxplot(BS ~ RiskLevel, data = data, names = c("<=Med", "High"), col = c("aquamarine3", "
  aquamarine4"), outpch = 24)
boxplot(BodyTemp ~ RiskLevel, data = data, names = c("<=Med", "High"), col = c("azure4", "
  azure4"), outpch = 24)
boxplot(HeartRate ~ RiskLevel, data = data, names = c("<=Med", "High"), col = c("darkorchid3
  ", "darkorchid4"), outpch = 24)

# Correlations
cor(data[which(data$RiskLevel==0),-c(7)]) #medium-low risk
cor(data[which(data$RiskLevel==1),-c(7)]) #high risk

# Univariate. Youden index
for(i in seq(1,6,1)){
  print(i)
  p<-prediction(as.numeric(data[,i]), data[,dim(data)[2]])
  at<-attributes(performance(p, "sens", "spec"))
  maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
  print(paste('youden', round(maxxx,3)))
}

```

C.2. Performance results

```

# Authors: Rocío Aznar-Gimeno, Luis M. Esteban, Gerardo Sanz, Rafael del-Hoyo-Alonso
# Description: This code calculates the mean of the maximum Youden indices, as well as
  the sensitivity and specificity, obtained after applying the logistic regression,
  xgboost algorithm, min-max approach, min-max-median approach and min-max-IQR
  approach, on the Maternal Health Risk dataset (high vs. medium-low risk).
# Dataset: The Maternal Health Risk dataset can be found at http://archive.ics.uci.edu/ml
# Imports needed: MASS, SLModels, caret, xgboost
# Citation: If you use this code, please cite the article 'Comparing the Min-Max-Median/
  IQR approach with Min-Max approach, Logistic Regression and XGBoost, Maximising the
  Youden Index', and reference the SLModels library.

library(MASS)
library(SLModels)
library(caret)
library(xgboost)

set.seed(2)

# Load dataset
datos <- read.csv("maternal.csv", sep=",", header=TRUE)
datos$RiskLevel<-unclass(datos$RiskLevel)
datos$RiskLevel <- replace(datos$RiskLevel, datos$RiskLevel=="high risk", 1)
datos$RiskLevel <- replace(datos$RiskLevel, datos$RiskLevel!=1, 0)
datos<-datos[which(datos$Age>=13 & datos$Age<=50 & datos$HeartRate>7),]
datos<-datos[!duplicated(datos),]
colnames(datos)<-c("V1", "V2", "V3", "V4", "V5", "V6", "z")
data<-datos
data<-data[
  with(data, order(z)),
]
n1<-dim(data[data$z == 0,])[1]
n2<-dim(data[data$z == 1,])[1]
data$z <- as.numeric(data$z)

data1<-data
folds <- createFolds(data1$z, k = 10)
data1 <- data.frame(data1$V1, data1$V2, data1$V3, data1$V4, data1$V5, data1$V6, data1$z)

data10<-data1
media1<-mean(data10[,1])
media2<-mean(data10[,2])
media3<-mean(data10[,3])
media4<-mean(data10[,4])
media5<-mean(data10[,5])
media6<-mean(data10[,6])

```

```

sd1<-sd(data10[,1])
sd2<-sd(data10[,2])
sd3<-sd(data10[,3])
sd4<-sd(data10[,4])
sd5<-sd(data10[,5])
sd6<-sd(data10[,6])

for (row in 1:dim(data1)[1]) {
  data10[row, 1] <- (data10[row, 1]-media1)/sd1
  data10[row, 2] <- (data10[row, 2]-media2)/sd2
  data10[row, 3] <- (data10[row, 3]-media3)/sd3
  data10[row, 4] <- (data10[row, 4]-media4)/sd4
  data10[row, 5] <- (data10[row, 5]-media5)/sd5
  data10[row, 6] <- (data10[row, 6]-media6)/sd6
}

youdenSummary <- function(data, lev = NULL, model = NULL){
  if (length(lev) > 2) {
    stop(paste("Your outcome has", length(lev), "levels. The youdenSummary() function isn't
      appropriate."))
  }
  if (!all(levels(data[, "pred"]) == lev)) {
    stop("levels of observed and predicted data do not match")
  }
  Sens <- caret::sensitivity(data[, "pred"], data[, "obs"], lev[1])
  Spec <- caret::specificity(data[, "pred"], data[, "obs"], lev[2])
  j <- Sens + Spec
  out <- c(j, Spec, Sens)
  names(out) <- c("j", "Spec", "Sens")
  out
}

youden_log <-c()
sensitivity_log<-c()
specificity_log<-c()
youden_xg <-c()
sensitivity_xg<-c()
specificity_xg<-c()
youden_mm <-c()
sensitivity_mm<-c()
specificity_mm<-c()
youden_mmm <-c()
sensitivity_mmm<-c()
specificity_mmm<-c()
auc_mmiqr<-c()
acc_mmiqr<-c()
youden_mmiqr <-c()
sensitivity_mmiqr<-c()
specificity_mmiqr<-c()

# 10-fold cross validation
for (ff in folds){

  # Split training and validation
  train <- data1[-ff,]
  valid <- data1[ff,]
  train<-train[,c(1,2,3,4,5,6,7)]
  valid<-valid[,c(1,2,3,4,5,6,7)]
  train0 <- data10[-ff,]
  valid0 <- data10[ff,]
  train0<-train0[,c(1,2,3,4,5,6,7)]
  valid0<-valid0[,c(1,2,3,4,5,6,7)]

  ## Training ##
  # Logistic
  logistic<-glm(data1.z ~ ., train,family="binomial")
  aa<-unlist(logistic$coefficients[2])*unlist(train[,1])+unlist(logistic$
    coefficients[3])*unlist(train[,2])+unlist(logistic$coefficients[4])*unlist(
    train[,3])+unlist(logistic$coefficients[5])*unlist(train[,4])+unlist(logistic
    $coefficients[6])*unlist(train[,5])+unlist(logistic$coefficients[7])*unlist(
    train[,6])
  data2<-matrix(aa)

```

```

p<-prediction(data2,train[,dim(train)[2]])
a<-attributes(performance(p,"sens","spec"))
cutofflog<-a$alpha.values[[1]][which.max(a$y.values[[1]]+a$x.values[[1]]-1)]
# XGBoost
train[, which(names(train) %in% c("data1.z"))]<-as.factor(train[, which(names(
train) %in% c("data1.z"))])
train_control = trainControl(method = "cv", number = 5, search = "grid",
summaryFunction = youdenSummary)
gbmGrid <- expand.grid(max_depth = c(2,3,6,8,10,20), nrounds = c(50,100,200), eta
= c(0.1, 0.3), gamma = c(0, 0.5), subsample = c(0.5, 1.0), min_child_weight =
1, colsample_bytree = c(0.5,1))
model0 = train(data1.z~., data = train, method = "xgbTree", metric="j", trControl
= train_control, tuneGrid = gbmGrid)
hipers <- model0$finalModel$tuneValue
train$data1.z<-as.numeric(as.character(train$data1.z))
model <- xgboost(data = data.matrix(train[, -which(names(train) %in% c("data1.z")
)]), label=as.numeric(train$data1.z), objective = "binary:logistic", early_
stopping_rounds = 10,nthread = 2, max_depth = hipers$max_depth, eta = hipers$
eta, nrounds = hipers$nrounds, gamma = hipers$gamma, colsample_bytree =
hipers$colsample_bytree, min_child_weight=hipers$min_child_weight, subsample=
hipers$subsample)
dataa<-xgb.DMatrix(data.matrix(train[, -which(names(train) %in% c("data1.z"))]),
label=train$data1.z)
predictions <- predict(model, dataa)
p<-prediction(predictions,train[,dim(train)[2]])
a<-attributes(performance(p,"sens","spec"))
cutoffxg<-a$alpha.values[[1]][which.max(a$y.values[[1]]+a$x.values[[1]]-1)]
# MM
a_mm<-SLModels(train0,algorithm="minmax")
# MMM
a_mmm<-SLModels(train0,algorithm="minmaxmedian")
# MMIQR
a_mmiqr<-SLModels(train0,algorithm="minmaxiqr")

## Validation ##
# Logistic
aa<-unlist(logistic$coefficients[2])*unlist(valid[,1])+unlist(logistic$
coefficients[3])*unlist(valid[,2])+unlist(logistic$coefficients[4])*unlist(
valid[,3])+unlist(logistic$coefficients[5])*unlist(valid[,4])+unlist(logistic
$coefficients[6])*unlist(valid[,5])+unlist(logistic$coefficients[7])*unlist(
valid[,6])
data22<-matrix(aa)
data222<-data22[,1]
data222 <- ifelse(data222<cutofflog, 0, 1)
p<-prediction(data222,valid[,dim(valid)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
youden_log<-c(youden_log,maxx)
sensitivity_log<-c(sensitivity_log,at$y.values[[1]][which.max(at$y.values[[1]]+at$x.
values[[1]]-1)])
specificity_log<-c(specificity_log,at$x.values[[1]][which.max(at$y.values[[1]]+at$x.
values[[1]]-1)])
# XGBoost
valid[, which(names(valid) %in% c("data1.z"))]<-as.factor(valid[, which(names(
valid) %in% c("data1.z"))])
valid$data1.z<-as.numeric(as.character(valid$data1.z))
dataa<-xgb.DMatrix(data.matrix(valid[, -which(names(valid) %in% c("data1.z"))]),
label=valid$data1.z)
predictions <- predict(model, dataa)
predictions <- ifelse(predictions<cutoffxg, 0, 1)
p<-prediction(predictions,valid[,dim(valid)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
youden_xg<-c(youden_xg,maxx)
sensitivity_xg<-c(sensitivity_xg,at$y.values[[1]][which.max(at$y.values[[1]]+at$x.
values[[1]]-1)])
specificity_xg<-c(specificity_xg,at$x.values[[1]][which.max(at$y.values[[1]]+at$x.
values[[1]]-1)])
# MM
aa<-cbind(apply(valid0[,c(1,2,3,4,5,6)],1,max),apply(valid0[,c(1,2,3,4,5,6)],1,min
))

```

```

data22<-matrix(unlist(a_mm['max'])*unlist(aa[,1])+unlist(a_mm['min'])*unlist(aa
[,2]))
data222<-data22[,1]
data222 <- ifelse(data222<unlist(a_mm['Cutoff']), 0, 1)
p<-prediction(data222,valid0[,dim(valid0)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
youden_mm<-c(youden_mm,maxx)
sensitivity_mm<-c(sensitivity_mm,at$y.values[[1]][which.max(at$y.values[[1]]+at$x.
values[[1]]-1)])
specificity_mm<-c(specificity_mm,at$x.values[[1]][which.max(at$y.values[[1]]+at$x.
values[[1]]-1)])
# MMM
aa<-cbind(apply(valid0[,c(1,2,3,4,5,6)],1,max),apply(valid0[,c(1,2,3,4,5,6)],1,min
),apply(valid0[,c(1,2,3,4,5,6)],1,median))
data22<-matrix(unlist(a_mmm['max'])*unlist(aa[,1])+unlist(a_mmm['min'])*unlist(aa
[,2])+unlist(a_mmm['median'])*unlist(aa[,3]))
data222<-data22[,1]
data222 <- ifelse(data222<unlist(a_mmm['Cutoff']), 0, 1)
p<-prediction(data222,valid0[,dim(valid0)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
youden_mmm<-c(youden_mmm,maxx)
sensitivity_mmm<-c(sensitivity_mmm,at$y.values[[1]][which.max(at$y.values[[1]]+at$x.
values[[1]]-1)])
specificity_mmm<-c(specificity_mmm,at$x.values[[1]][which.max(at$y.values[[1]]+at$x.
values[[1]]-1)])
# MMIQR
aa<-cbind(apply(valid0[,c(1,2,3,4,5,6)],1,max),apply(valid0[,c(1,2,3,4,5,6)],1,min
),apply(valid0[,c(1,2,3,4,5,6)],1,quantile)[3,]-apply(valid0[,c(1,2,3,4,5,6)
],1,quantile)[1,])
data22<-matrix(unlist(a_mmiqr['max'])*unlist(aa[,1])+unlist(a_mmiqr['min'])*unlist
(aa[,2])+unlist(a_mmiqr['iqr'])*unlist(aa[,3]))
data222<-data22[,1]
data222 <- ifelse(data222<unlist(a_mmiqr['Cutoff']), 0, 1)
p<-prediction(data222,valid0[,dim(valid0)[2]])
at<-attributes(performance(p,"sens","spec"))
maxx<-max(at$y.values[[1]]+at$x.values[[1]]-1)
youden_mmiqr<-c(youden_mmiqr,maxx)
sensitivity_mmiqr<-c(sensitivity_mmiqr,at$y.values[[1]][which.max(at$y.values[[1]]+at
$x.values[[1]]-1)])
specificity_mmiqr<-c(specificity_mmiqr,at$x.values[[1]][which.max(at$y.values[[1]]+at
$x.values[[1]]-1)])
}

# Final results
config<-c("logistic", "xgboost","mm", "mmm", "mmiqr")
mean_youden_log<-c(mean(youden_log),mean(youden_xg), mean(youden_mm), mean(youden_mmm),
mean(youden_mmiqr))
mean_sensitivity_log<-c(mean(sensitivity_log),mean(sensitivity_xg), mean(sensitivity_mm),
mean(sensitivity_mmm), mean(sensitivity_mmiqr))
mean_specificity_log<-c(mean(specificity_log),mean(specificity_xg), mean(specificity_mm),
mean(specificity_mmm), mean(specificity_mmiqr))
tabla <- data.frame(config, mean_youden_log,mean_sensitivity_log,mean_specificity_log)

```