OPEN ACCESS protocols.io

Apr 05, 2019

Working

Prediction of pneumoconiosis by serum and urinary biomarkers in workers exposed to asbestos-contaminated minerals

PLOS One

Hsiao-Yu Yang[1]

[1]National Taiwan University

dx.doi.org/10.17504/protocols.io.x8dfrs6

Hsiao-Yu Yang
National Taiwan University

ABSTRACT

Prediction of pneumoconiosis by serum and urinary biomarkers in workers exposed to asbestos-contaminated minerals
PLOS One

EXTERNAL LINK

https://doi.org/10.1371/journal.pone.0214808

PROTOCOL STATUS

Working

SAFETY WARNINGS

Required software

1 This protocol requires R-3.5.2 for Windows (32/64 bit) (https://cran.r-project.org/bin/windows/base/)

Dataset preparation

2 Download the dataset and store at "C:\r". **PLoS One Excel Main Raw Datafile.csv**

Cpoy the R script of six machine learning algorithms on R consol.

3

```
library(rattle)   # Access the weather dataset and utilities.
library(magrittr) # Utilise %>% and %<>% pipeline operators.

building <- TRUE
scoring  <- ! building

# A pre-defined value is used to reset the random seed
# so that results are repeatable.

crv$seed <- 42

# Load a dataset from file.

fname     <- "file:///C:/r/PLoS One Excel Main Raw Datafile.csv"
crs$dataset <- read.csv(fname,
  na.strings=c(".", "NA", "", "?"),
  strip.white=TRUE, encoding="UTF-8")


# Action the user selections from the Data tab.

# The following variable selections have been noted.

crs$input     <- c("Age", "Sex", "CEA", "SMRP", "Fibulin3",
                   "OhdG")

crs$numeric   <- c("Age", "Sex", "CEA", "SMRP", "Fibulin3",
                   "OhdG")

crs$categoric <- NULL

crs$target    <- "Pneumoconiosis"
crs$risk      <- NULL
crs$ident     <- NULL
crs$ignore    <- c("FeNO", "FVC", "FEV1", "Smoking")
crs$weights   <- NULL

#============================================================
# Decision Tree

# The 'rpart' package provides the 'rpart' function.

library(rpart, quietly=TRUE)

# Reset the random number seed to obtain the same results each time.

set.seed(crv$seed)

# Build the Decision Tree model.

crs$rpart <- rpart(Pneumoconiosis ~ .,
    data=crs$dataset[, c(crs$input, crs$target)],
    method="class",
    parms=list(split="information"),
    control=rpart.control(usesurrogate=0,
        maxsurrogate=0),
    model=TRUE)

# Generate a textual view of the Decision Tree model.

print(crs$rpart)
printcp(crs$rpart)
cat("\n")

#
#============================================================
# Extreme Boost

# The `xgboost' package implements the extreme gradient boost algorithm.

# Build the Extreme Boost model.

set.seed(crv$seed)
```

```
crs$ada <- xgboost(Pneumoconiosis ~ .,
  data             = crs$dataset[,c(crs$input, crs$target)],
  max_depth        = 6,
  eta              = 0.3,
  num_parallel_tree = 1,
  nthread          = 2,
  nround           = 50,
  metrics          = 'error',
  objective        = 'binary:logistic')

# Print the results of the modelling.

print(crs$ada)

cat('\nFinal iteration error rate:\n')
print(round(crs$ada$evaluation_log[crs$ada$niter, ], 2))

cat('\nImportance/Frequency of variables actually used:\n')
print(crs$imp <- importance(crs$ada, crs$dataset[,c(crs$input, crs$target)]))


#=================================================================
# Build a Random Forest model using the traditional approach.

set.seed(crv$seed)

crs$rf <- randomForest::randomForest(as.factor(Pneumoconiosis) ~ .,
  data=crs$dataset[, c(crs$input, crs$target)],
  ntree=500,
  mtry=2,
  importance=TRUE,
  na.action=randomForest::na.roughfix,
  replace=FALSE)

# Generate textual output of the 'Random Forest' model.

crs$rf

# The `pROC' package implements various AUC functions.

# Calculate the Area Under the Curve (AUC).

pROC::roc(crs$rf$y, as.numeric(crs$rf$predicted))

# Calculate the AUC Confidence Interval.

pROC::ci.auc(crs$rf$y, as.numeric(crs$rf$predicted))FALSE

# List the importance of the variables.

rn <- round(randomForest::importance(crs$rf), 2)
rn[order(rn[,3], decreasing=TRUE),]



#=================================================================
# Support vector machine.
# The 'kernlab' package provides the 'ksvm' function.

library(kernlab, quietly=TRUE)

# Build a Support Vector Machine model.

set.seed(crv$seed)
crs$ksvm <- ksvm(as.factor(Pneumoconiosis) ~ .,
      data=crs$dataset[,c(crs$input, crs$target)],
      kernel="rbfdot",
      prob.model=TRUE)

# Generate a textual view of the SVM model.

crs$ksvm



#============================================================
# Regression model

# Build a Regression model.

crs$glm <- glm(Pneumoconiosis ~ .,
    data=crs$dataset[, c(crs$input, crs$target)],
    family=binomial(link="logit"))

# Generate a textual view of the Linear model.

print(summary(crs$glm))

cat(sprintf("Log likelihood: %.3f (%d df)\n",
            logLik(crs$glm)[1],
            attr(logLik(crs$glm), "df")))

cat(sprintf("Null/Residual deviance difference: %.3f (%d df)\n",
            crs$glm$null.deviance-crs$glm$deviance,
            crs$glm$df.null-crs$glm$df.residual))

cat(sprintf("Chi-square p-value: %.8f\n",
            dchisq(crs$glm$null.deviance-crs$glm$deviance,
                   crs$glm$df.null-crs$glm$df.residual)))

cat(sprintf("Pseudo R-Square (optimistic): %.8f\n",
             cor(crs$glm$y, crs$glm$fitted.values)))

cat('\n==== ANOVA ====\n\n')
print(anova(crs$glm, test="Chisq"))
cat("\n")



#============================================================
# Neural Network

# Build a neural network model using the nnet package.

library(nnet, quietly=TRUE)
```

```
# Build the NNet model.

set.seed(199)
crs$nnet <- nnet(as.factor(Pneumoconiosis) ~ .,
    data=crs$dataset[,c(crs$input, crs$target)],
    size=10, skip=TRUE, MaxNWts=10000, trace=FALSE, maxit=100)

# Print the results of the modelling.

cat(sprintf("A %s network with %d weights.\n",
    paste(crs$nnet$n, collapse="-"),
    length(crs$nnet$wts)))
cat(sprintf("Inputs: %s.\n",
    paste(crs$nnet$coefnames, collapse=", ")))
cat(sprintf("Output: %s.\n",
    names(attr(crs$nnet$terms, "dataClasses"))[1]))
cat(sprintf("Sum of Squares Residuals: %.4f.\n",
    sum(residuals(crs$nnet) ^ 2)))
cat("\n")
print(summary(crs$nnet))
cat('\n')



#============================================================
# Evaluate model performance on the training dataset.

# Sensitivity/Specificity Plot: requires the ROCR package

library(ROCR)

# Generate Sensitivity/Specificity Plot for rpart model on PLoS One Excel Main Raw Datafile.csv [**train**].

crs$pr <- predict(crs$rpart, newdata=crs$dataset[,c(crs$input, crs$target)])[,2]

# Remove observations with missing target.

no.miss   <- na.omit(crs$dataset[,c(crs$input, crs$target)]$Pneumoconiosis)
miss.list <- attr(no.miss, "na.action")
attributes(no.miss) <- NULL

if (length(miss.list))
{
  pred <- prediction(crs$pr[-miss.list], no.miss)
} else
{
  pred <- prediction(crs$pr, no.miss)
}
ROCR::plot(performance(pred, "sens", "spec"), col="#CC0000FF", lty=1, add=FALSE)


# Sensitivity/Specificity Plot: requires the ROCR package

library(ROCR)

# Generate Sensitivity/Specificity Plot for xgb model on PLoS One Excel Main Raw Datafile.csv [**train**].

crs$pr <- predict(crs$ada, crs$dataset[,c(crs$input, crs$target)])
```

```
# Remove observations with missing target.

no.miss   <- na.omit(crs$dataset[,c(crs$input, crs$target)]$Pneumoconiosis)
miss.list <- attr(no.miss, "na.action")
attributes(no.miss) <- NULL

if (length(miss.list))
{
  pred <- prediction(crs$pr[-miss.list], no.miss)
} else
{
  pred <- prediction(crs$pr, no.miss)
}
ROCR::plot(performance(pred, "sens", "spec"), col="#CCCC00FF", lty=2, add=TRUE)


# Sensitivity/Specificity Plot: requires the ROCR package

library(ROCR)

# Generate Sensitivity/Specificity Plot for rf model on PLoS One Excel Main Raw Datafile.csv [**train**].

crs$pr <- predict(crs$rf, newdata=na.omit(crs$dataset[,c(crs$input, crs$target)]),
    type    = "prob")[,2]

# Remove observations with missing target.

no.miss   <- na.omit(na.omit(crs$dataset[,c(crs$input, crs$target)])$Pneumoconiosis)
miss.list <- attr(no.miss, "na.action")
attributes(no.miss) <- NULL

if (length(miss.list))
{
  pred <- prediction(crs$pr[-miss.list], no.miss)
} else
{
  pred <- prediction(crs$pr, no.miss)
}
ROCR::plot(performance(pred, "sens", "spec"), col="#00CC00FF", lty=3, add=TRUE)


# Sensitivity/Specificity Plot: requires the ROCR package

library(ROCR)

# Generate Sensitivity/Specificity Plot for ksvm model on PLoS One Excel Main Raw Datafile.csv [**train**].

crs$pr <- kernlab::predict(crs$ksvm, newdata=na.omit(crs$dataset[,c(crs$input, crs$target)]),
    type    = "probabilities")[,2]

# Remove observations with missing target.

no.miss   <- na.omit(na.omit(crs$dataset[,c(crs$input, crs$target)])$Pneumoconiosis)
miss.list <- attr(no.miss, "na.action")
attributes(no.miss) <- NULL

if (length(miss.list))
{
  pred <- prediction(crs$pr[-miss.list], no.miss)
} else
{
  pred <- prediction(crs$pr, no.miss)
}
ROCR::plot(performance(pred, "sens", "spec"), col="#00CCCCFF", lty=4, add=TRUE)


# Sensitivity/Specificity Plot: requires the ROCR package

library(ROCR)

# Generate Sensitivity/Specificity Plot for glm model on PLoS One Excel Main Raw Datafile.csv [**train**].

crs$pr <- predict(crs$glm,
   type    = "response",
   newdata = crs$dataset[,c(crs$input, crs$target)])

# Remove observations with missing target.

no.miss   <- na.omit(crs$dataset[,c(crs$input, crs$target)]$Pneumoconiosis)
miss.list <- attr(no.miss, "na.action")
attributes(no.miss) <- NULL

if (length(miss.list))
{
  pred <- prediction(crs$pr[-miss.list], no.miss)
} else
{
  pred <- prediction(crs$pr, no.miss)
}
ROCR::plot(performance(pred, "sens", "spec"), col="#0000CCFF", lty=5, add=TRUE)


# Sensitivity/Specificity Plot: requires the ROCR package

library(ROCR)

# Generate Sensitivity/Specificity Plot for nnet model on PLoS One Excel Main Raw Datafile.csv [**train**].

crs$pr <- predict(crs$nnet, newdata=crs$dataset[,c(crs$input, crs$target)])

# Remove observations with missing target.

no.miss   <- na.omit(crs$dataset[,c(crs$input, crs$target)]$Pneumoconiosis)
miss.list <- attr(no.miss, "na.action")
attributes(no.miss) <- NULL

if (length(miss.list))
{
  pred <- prediction(crs$pr[-miss.list], no.miss)
} else
{
  pred <- prediction(crs$pr, no.miss)
}
ROCR::plot(performance(pred, "sens", "spec"), col="#CC00CCFF", lty=6, add=TRUE)


# Add a legend to the plot.
```

```
legend("bottomleft", c("rpart","xgb","rf","ksvm","glm","nnet"), col=rainbow(6, 1, .8), lty=1:6, title="Models", inset=c(0.(
5))

# Add decorations to the plot.

title(main="Sensitivity/Specificity (tpr/tnr)  PLoS One Excel Main Raw Datafile.csv [**train**]",
    sub=paste("Rattle", format(Sys.time(), "%Y-%b-%d %H:%M:%S"), Sys.info()["user"]))
grid()
```