#@Author :Sai Kian Ippili

#@Title : Elastic Multi Stage Decision Methodology

#@Date : 03/03/2017 – 05/08/2017

#packages

# install.packages("rpart")

# install.packages("rpart.plot")

# install.packages("partykit")

# install.packages("Metrics")

# install.packages("gWidgets2")

# install.packages("gWidgets2tcltk")

#loading libraries

library(rpart)

library(rpart.plot)

library(partykit)

library(Metrics)

require(gWidgets2)

require(gWidgets2tcltk)

# #Dataset

EMSDM\_dataset <- read.csv(file.choose(),header = TRUE)

EMSDM\_formula <- EMSDM\_dataset$play ~ .

#function call for Building Tree

EMSDM\_Window(EMSDM\_formula,EMSDM\_dataset)

#function definition for Building Tree

EMSDM\_Window <- function(EMSDM\_formula,EMSDM\_dataset)

{

#position of dependent variable in dataset

EMSDM\_y <- as.character(EMSDM\_formula)

EMSDM\_y <- substr(EMSDM\_y[2], 15, nchar(EMSDM\_y[2]))

EMSDM\_y\_position <<- grep(EMSDM\_y, colnames(EMSDM\_dataset))

#Window

w <- gwindow("gformlayout", visible = FALSE)

g <- gvbox(container = w)

flyt <- gformlayout(container = g)

i <<- -1

b <-

gbutton(

"Show me next split",

container = g,

handler = function(h, ...) {

#print(svalue(flyt))

#implementing the rpart algorithm

i <<- i + 1

if (i == 0)

{

fit <- rpart(EMSDM\_formula,

data = EMSDM\_dataset,

method = "class",

cp = 100)

rpart.plot(fit)

prp(fit, main="Elastic Multi-Stage Decision Methodology",

type=4, fallen=T, branch=.3, round=0, leaf.round=9,

clip.right.labs=F, under.cex=1,

box.palette="GnYlRd",

prefix="", branch.col="gray", branch.lwd=2,

extra=101, under=T, lt=" < ", ge=" >= ", cex.main=1)

EMSDM\_Metrics(fit, EMSDM\_dataset[1:(ncol(EMSDM\_dataset))])

}

else

{

fit <- rpart(

EMSDM\_formula,

data = EMSDM\_dataset,

method = "class",

maxdepth = i,

minsplit = 2,

minbucket = 1 ,cp = -1

)

#rpart.plot(fit,type=4,nn = TRUE,under.cex=3,main="Elastic Multi-Stage Decision Methodology")

# rpart.plot(binary.model, # middle graph

# extra=104, box.palette="GnBu",

# branch.lty=3, shadow.col="gray", nn=TRUE)

#

prp(fit, main="Elastic Multi-Stage Decision Methodology",

type=4, fallen=T, branch=.3, round=0, leaf.round=9,

clip.right.labs=F, under.cex=1,

box.palette="GnYlRd",

prefix="", branch.col="gray",nn = TRUE ,branch.lwd=2,

extra=101, under=T, lt=" < ", ge=" >= ", cex.main=1,cex = 0.8)

EMSDM\_node\_observations(fit, i, EMSDM\_dataset)

}

print(i)

}

)

bprune <-

gbutton(

"Prune",

container = g,

handler = function(h, ...) {

#print(svalue(flyt))

#implementing the rpart algorithm

if (i <= 1)

{

fit <- rpart(EMSDM\_formula,

data = EMSDM\_dataset,

method = "class",

cp = 100)

rpart.plot(fit)

prp(fit, main="Elastic Multi-Stage Decision Methodology",

type=4, fallen=T, branch=.3, round=0, leaf.round=9,

clip.right.labs=F, under.cex=1,

box.palette="GnYlRd",

prefix="", branch.col="gray", branch.lwd=2,

extra=101, under=T, lt=" < ", ge=" >= ", cex.main=1)

EMSDM\_Metrics(fit, EMSDM\_dataset[1:(ncol(EMSDM\_dataset))])

if(i==1)

{

i <<- i - 1

}

}

else

{

i <<- i - 1

fit <- rpart(

EMSDM\_formula,

data = EMSDM\_dataset,

method = "class",

maxdepth = i,

minsplit = 2,

minbucket = 1, cp = -1

)

#rpart.plot(fit,type=4,nn = TRUE,under.cex=3,main="Elastic Multi-Stage Decision Methodology")

# rpart.plot(binary.model, # middle graph

# extra=104, box.palette="GnBu",

# branch.lty=3, shadow.col="gray", nn=TRUE)

#

prp(fit, main="Elastic Multi-Stage Decision Methodology",

type=4, fallen=T, branch=.3, round=0, leaf.round=9,

clip.right.labs=F, under.cex=1,

box.palette="GnYlRd",

prefix="", branch.col="gray",nn = TRUE ,branch.lwd=2,

extra=101, under=T, lt=" < ", ge=" >= ", cex.main=1,cex = 0.8)

EMSDM\_node\_observations(fit, i, EMSDM\_dataset)

}

print(i)

}

)

addSpring(g) ## better with Qt, else flyt expands to fill.

visible(w) <- TRUE

}

## End(Not run)

#summary(fit)

#plotting decition tree

#calculate metrics

EMSDM\_Metrics <- function(fit, data\_metrics)

{

if (i == 0)

{

predict\_data <-

predict(fit, data\_metrics[1:nrow(data\_metrics),], type = "class")

comparisionmatrix <-

table(data\_metrics[1:nrow(data\_metrics),EMSDM\_y\_position], predicted =

predict\_data)

}

else{

predict\_data <-

predict(fit, data\_metrics[1:nrow(data\_metrics),], type = "class")

comparisionmatrix <-

table(data\_metrics[1:nrow(data\_metrics), EMSDM\_y\_position], predicted = predict\_data)

}

# prepare resampling method

n <- sum(comparisionmatrix) # number of instances

nc <- nrow(comparisionmatrix) # number of classes

# number of correctly classified instances per class

diagonal <- diag(comparisionmatrix)

row\_sums = apply(comparisionmatrix, 1, sum) # number of instances per class

col\_sums = apply(comparisionmatrix, 2, sum) # number of predictions per class

p = row\_sums / n # distribution of instances over the actual classes

q = col\_sums / n # distribution of instances over the predicted classes

#RMSE <- sqrt(mean((y-predict\_data)^2))

#Accuracy

accuracy <- sum(diagonal) / n

#Per-class Precision, Recall, and F-1

precision = diagonal / col\_sums

recall = diagonal / row\_sums

f1 = 2 \* precision \* recall / (precision + recall)

PRF1 <- data.frame(precision, recall, f1)

#Kappa Statistic

expAccuracy = sum(p \* q)

kappa = (accuracy - expAccuracy) / (1 - expAccuracy)

print(paste0("Number of instances :", n))

print(paste0(

"Correctly Classified Instances :",

sum(diagonal) ,

" ",

(sum(diagonal) / n) \* 100,

"%"

))

print(paste0(

"InCorrectly Classified Instances :",

n - sum(diagonal) ,

" ",

((n - sum(diagonal)) / n) \* 100,

"%"

))

print(paste0("Comparision Matrix :"))

print(comparisionmatrix)

print(paste0("Accuracy :", accuracy\*100,"%"))

print("Per-class Precision, Recall, and F-1 :")

print(PRF1)

print(paste0("Kappa :", kappa))

return()

}

EMSDM\_node\_observations <- function(fit, i, EMSDM\_dataset)

{

#To get observations at each node

pfit <- as.party(fit)

#pfit[[1]]$ node

#plot(pfit)

#pfit$data <- model.frame(fit)

if(length(pfit) > 0)

{

EMSDM\_Metrics(fit, EMSDM\_dataset)

}

}