รายงานผลการดำเนินงานสร้าง Decision Tree Model สำหรับชุดข้อมูล Mushroom

จาก UCI Machine Learning repository

2.1 การเตรียมชุดข้อมูล (Data acquisition)

อ่านไฟล์ด้วยคำสั่ง read.csv และทำการดูข้อมูลในไฟล์ มีทั้งหมด 23 Feature/variable

Code:

mushroomData <- read.csv("mushrooms.csv")

str(mushroomData)

summary(mushroomData)

ผลลัพธ์การดูข้อมูลในไฟล์ mushrooms.csv มี23 variables มีจำนวนข้อมูลทั้งหมด 8124 ข้อมูล ข้อมูvariables ที่ใช้เป็น Target value คือ class มี2 levels คือ e = edible และ p= poisonous ดังนั้น variables ที่เหลือจะเป็น Feature ในการมาทำโมเดล ว่าเห็ดลักษณะใดเป็น edible หรือ poisonous ทำการ summaryข้อมูลเพื่อดูภาพรวมตำแหน่งข้อมูลต่างๆ ว่าแต่ละ Level มีข้อมูลเท่าไร

ข้อมูลที่ได้จากการปริ้นข้อมูลในดาต้า

'data.frame': 8124 obs. of 23 variables:

$ class : Factor w/ 2 levels "e","p": 2 1 1 2 1 1 1 1 2 1 ...

$ cap.shape : Factor w/ 6 levels "b","c","f","k",..: 6 6 1 6 6 6 1 1 6 1 ...

$ cap.surface : Factor w/ 4 levels "f","g","s","y": 3 3 3 4 3 4 3 4 4 3 ...

$ cap.color : Factor w/ 10 levels "b","c","e","g",..: 5 10 9 9 4 10 9 9 9 10 ...

$ bruises : Factor w/ 2 levels "f","t": 2 2 2 2 1 2 2 2 2 2 ...

$ odor : Factor w/ 9 levels "a","c","f","l",..: 7 1 4 7 6 1 1 4 7 1 ...

$ gill.attachment : Factor w/ 2 levels "a","f": 2 2 2 2 2 2 2 2 2 2 ...

$ gill.spacing : Factor w/ 2 levels "c","w": 1 1 1 1 2 1 1 1 1 1 ...

$ gill.size : Factor w/ 2 levels "b","n": 2 1 1 2 1 1 1 1 2 1 ...

$ gill.color : Factor w/ 12 levels "b","e","g","h",..: 5 5 6 6 5 6 3 6 8 3 ...

$ stalk.shape : Factor w/ 2 levels "e","t": 1 1 1 1 2 1 1 1 1 1 ...

$ stalk.root : Factor w/ 5 levels "?","b","c","e",..: 4 3 3 4 4 3 3 3 4 3 ...

$ stalk.surface.above.ring: Factor w/ 4 levels "f","k","s","y": 3 3 3 3 3 3 3 3 3 3 ...

$ stalk.surface.below.ring: Factor w/ 4 levels "f","k","s","y": 3 3 3 3 3 3 3 3 3 3 ...

$ stalk.color.above.ring : Factor w/ 9 levels "b","c","e","g",..: 8 8 8 8 8 8 8 8 8 8 ...

$ stalk.color.below.ring : Factor w/ 9 levels "b","c","e","g",..: 8 8 8 8 8 8 8 8 8 8 ...

$ veil.type : Factor w/ 1 level "p": 1 1 1 1 1 1 1 1 1 1 ...

$ veil.color : Factor w/ 4 levels "n","o","w","y": 3 3 3 3 3 3 3 3 3 3 ...

$ ring.number : Factor w/ 3 levels "n","o","t": 2 2 2 2 2 2 2 2 2 2 ...

$ ring.type : Factor w/ 5 levels "e","f","l","n",..: 5 5 5 5 1 5 5 5 5 5 ...

$ spore.print.color : Factor w/ 9 levels "b","h","k","n",..: 3 4 4 3 4 3 3 4 3 3 ...

$ population : Factor w/ 6 levels "a","c","n","s",..: 4 3 3 4 1 3 3 4 5 4 ...

$ habitat : Factor w/ 7 levels "d","g","l","m",..: 6 2 4 6 2 2 4 4 2 4 ...

ข้อมูลจากการ Summary ข้อมูล

class cap.shape cap.surface cap.color bruises odor gill.attachment gill.spacing

e:4208 b: 452 f:2320 n :2284 f:4748 n :3528 a: 210 c:6812

p:3916 c: 4 g: 4 g :1840 t:3376 f :2160 f:7914 w:1312

f:3152 s:2556 e :1500 s : 576

k: 828 y:3244 y :1072 y : 576

s: 32 w :1040 a : 400

x:3656 b : 168 l : 400

(Other): 220 (Other): 484

gill.size gill.color stalk.shape stalk.root stalk.surface.above.ring stalk.surface.below.ring

b:5612 b :1728 e:3516 ?:2480 f: 552 f: 600

n:2512 p :1492 t:4608 b:3776 k:2372 k:2304

w :1202 c: 556 s:5176 s:4936

n :1048 e:1120 y: 24 y: 284

g : 752 r: 192

h : 732

(Other):1170

stalk.color.above.ring stalk.color.below.ring veil.type veil.color ring.number ring.type

w :4464 w :4384 p:8124 n: 96 n: 36 e:2776

p :1872 p :1872 o: 96 o:7488 f: 48

g : 576 g : 576 w:7924 t: 600 l:1296

n : 448 n : 512 y: 8 n: 36

b : 432 b : 432 p:3968

o : 192 o : 192

(Other): 140 (Other): 156

spore.print.color population habitat

w :2388 a: 384 d:3148

n :1968 c: 340 g:2148

k :1872 n: 400 l: 832

h :1632 s:1248 m: 292

r : 72 v:4040 p:1144

b : 48 y:1712 u: 368

(Other): 144 w: 192

2.2. การแบ่งข้อมูลเพื่อ Train และ Test แบบจำลอง (Data partitioning)

แบ่งข้อมูลที่ใช้สำหรับ train 80% เพราะตามหลักการควรจะแบ่งข้อมูลในการtrain มากกว่าข้อมูลในการเทส และ ข้อมูล test 20%

set.seed(12345) # for reproducibility

train <- sample(1:nrow(mushrooms),size = ceiling(0.80\*nrow(mushrooms)))

# training set

mushrooms\_train <- mushrooms[train,]

# test set

mushrooms\_test <- mushrooms[-train,]

2.3. การเลือก Attribute เพื่อสร้างแบบจำลอง (Attribute selection)

จะทำการหาค่า Information gain แต่ละ variable ยกเว้น class variable เพื่อใช้เป็นค่าน้ำหนักในการคัดเลือกฟีเจอร์ โดยเพื่อจะทำการเลือก attribute ที่มีค่า information gain สูงที่สุดจะเป็นตัวสำคัญในสร้างแบบจำลองโดยในที่นี้ ใช้ function สำเร็จรูปในการ หา information gain

gainRatioResult <- GainRatioAttributeEval( class ~ . , data = mushrooms\_train )

print( sort( gainRatioResult, decreasing = TRUE ))

ผลที่ได้ จะเห็นได้ว่า attribute odor มีค่า information gain สูงสุดจึงนำ odor มาเป็นตัวสร้าง Model ต่อไป

odor gill.size stalk.surface.above.ring spore.print.color ring.type

0.390405856 0.259494661 0.233252515 0.219820959 0.207677946

bruises stalk.surface.below.ring gill.spacing gill.color stalk.color.above.ring

0.200268581 0.193718575 0.151752775 0.138409494 0.130607240

stalk.color.below.ring veil.color population ring.number gill.attachment

0.121139032 0.121129760 0.098977772 0.088701316 0.085622647

stalk.root habitat cap.shape cap.surface cap.color

0.074780955 0.067745831 0.028900979 0.018647627 0.014254015

stalk.shape veil.type

0.007731006 0.000000000

2.4. การแสดงภาพเกี่ยวกับ Attribute ที่เลือก (Attribute visualization)

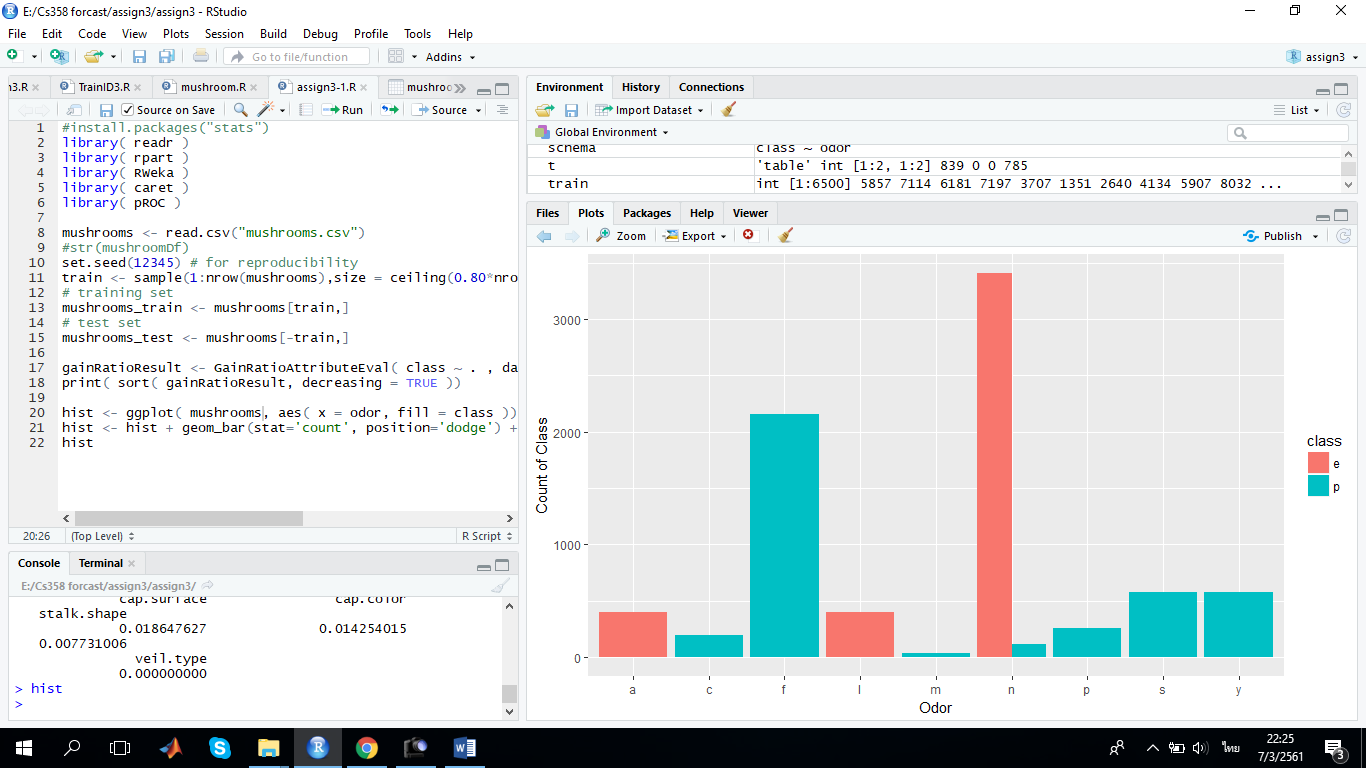
Ans นำข้อมูลใน variable odor มา plot graph เพื่อดูว่าข้อมูลใดใน level ของ odor ที่มีแนวโน้มจะเป็น poisonous หรือ edible ซึ่งจะทำให้สามารถจัดกลุ่มของข้อมูลได้ดี

hist <- ggplot( mushrooms, aes( x = odor, fill = class ))

hist <- hist + geom\_bar(stat='count', position='dodge') + labs(x = 'Odor', y = 'Count of Class')

hist

ผลลัพธ์ที่ได้จะแสดง ค่าที่มีโอกาสเป็นp= poisonous คือ c,f,m,p,s,y ค่าที่มีโอกาสเป็น e=edible คือ a,i และค่าที่มีโอกาสเป็นทั้ง poisonous และ edible คือ n



2.5. Classification ด้วย Decision Tree (Classification with Decision Tree) แบบที่ได้เรียนมาในชม.บรรยาย

2.5.1. ใช้วิธีการเขียนฟังก์ชันต่างๆ เพื่อสร้าง decision tree เอง อาศัย data.tree package (อ้างอิงจาก Lab)

อธิบายขั้นตอนการสร้างแบบจำลองด้วยวิธีการแบบที่ทำใน lab แนบ code ประกอบ และแสดง decision tree

ที่ได้ รวมทั้งผลการทดสอบด้วย

Ans การสร้างแบบจำลอง

1. ทำการอ่านข้อมูลเข้ามาเก็บไว้ที่ตัวแปร mushrooms ดูข้อมูลในดาต้าประกอบด้วย 23 variable ดูค่าที่เป็น Target value ที่เราต้องการสร้างโมเดล

2. ทำการตรวจสอบว่าข้อมูลนั้น pure หรือไม่ เพื่อให้สามารถเลือก attribute ที่จะนำมาทำ Model ได้ถูกต้องด้วยfunction IsPure

3. เพื่อที่ดูความแตกต่างของข้อมูลว่า มีการกระจายข้อมูลแบบใด ต้องคำนวณโดยใช้ค่า Entropy ถ้าให้ค่าน้อย = กระจายข้อมูลน้อย ในที่นี้จะดูการกระจายข้อมูลว่า e และ p ของ variable class กระจายข้อมูลเป็นอย่างไร

4. เพื่อเป็นการเลือก attribute ที่มีความสำคัญในการเป็นฟีเจอร์เพื่อสร้างแบบจำลอง attribute ที่มีค่า information gain สูงที่สุดจะเป็นตัวสำคัญในสร้างแบบจำลอง ในข้อมูลนี้ค่าที่มี InformationGain สูงที่สุดคือ odor = 0.906075

5. ในการสร้าง model decision tree จะต้องทำการเช็คของมูลก่อนว่า pure โดยใช้ fuction Ispure ถ้าหากค่าเป็น FALSE จะทำการหา information gain ของแต่ละ variable และจะเก็บค่าที่มีค่าสูงสุดมาทำเป็นตัวแรกในการทำModel ต่อไปเรื่อยๆ ทำการสร้างtree เมื่อมี node เข้ามาก็จะทำการสร้าง Decision Tree ขึ้นมา

6. สร้าง function Prediction เพื่อทำการทดสอบว่า Modelที่สร้างสามารถทำนายข้อมูลได้ถูกต้องหรือไม่

Code:

library(data.tree)

library( readr )

mushrooms <- read.csv("mushrooms.csv")

IsPure<-function(data){

length(unique(data[,1])) == 1

}

Entropy<-function(vl){

en <- vl/sum(vl) \* log2(vl/sum(vl))

en[vl == 0] <-0 #asign Nan

-sum(en)

}

InformationGain<-function(tble){

tble<- as.data.frame.matrix(tble)

enBefore<- Entropy(colSums(tble))

s <- rowSums(tble)

enAfter <- sum(s/sum(s)\*apply(tble, MARGIN = 1, FUN = Entropy))#apply = for loop

informationGain <- enBefore - enAfter

return(informationGain)

}

print(InformationGain(table(mushrooms[,c('odor','class')])))

TrainID3 <- function(node, data) {

node$obsCount <- nrow(data)

if (IsPure(data)) {

child <- node$AddChild(unique(data[,1]))

node$feature <- tail(names(data), 1)

child$obsCount <- nrow(data)

child$feature <- ''

} else {

ig <- sapply(colnames(data)[-1],

function(x) InformationGain(

table(data[,x], data[,1])

)

)

feature <- names(ig)[ig == max(ig)][1]

node$feature <- feature

childObs <- split(data[,!(names(data) %in% feature)],

data[,feature], drop = TRUE)

for(i in 1:length(childObs)) {

child <- node$AddChild(names(childObs)[i])

TrainID3(child, childObs[[i]])

}

}

}

Predict <- function(tree,feature){

if(tree$children[[1]]$isLeaf)

return(tree$children[[1]]$name)

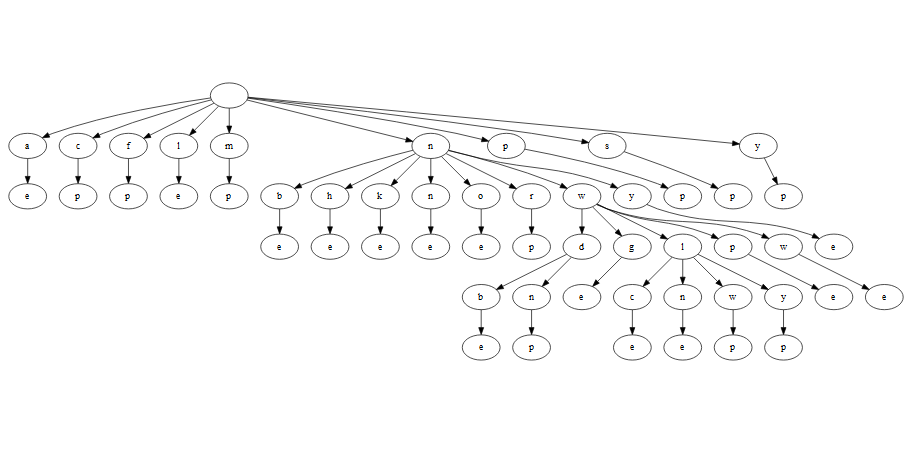
child <- tree$children[[feature[[tree$feature]]]]

return(Predict(child,feature))

} -sum(en)

}

Decision Tree:



ผลการทดสอบ ทำการใส่ข้อมูลใน function Prediction

Predict(tree, c(cap.shape='x', cap.surface='s', cap.color='n', bruises='t', odor='p', gill.attachment='NA',

gill.spacing='c', gill.size='n', gill.color='k', stalk.shape='e', stalk.root='NA', stalk.surface.above.ring='NA', stalk.surface.below.ring='s', talk.color.above.ring='w', stalk.color.below.ring='w', veil.type='p', veil.color='w', ring.number='o',

ring.type='p', spore.print.color='k', population='s', habitat='u'))

ผลการทดสอบ ได้ p

2.5.2. ใช้วิธีการเรียกไลบราลีสำเร็จรูปในการสร้าง decision tree เช่น rpart package

อธิบายขั้นตอนการสร้างแบบจำลองด้วยการเรียกใช้ฟังก์ชันในไลบราลีสำเร็จรูป แนบ code ประกอบ อธิบาย

ที่มาของค่าพารามิเตอร์ต่างๆ ที่ใช้ และแสดง decision tree ที่ได้ รวมทั้งผลการทดสอบด้วยAns ก่อนเรียกใช้ฟังก์ชันในไลบารี rpart ต้องทำการแบ่งข้อมูลเพื่อ Train และ Test แบบจำลอง ก็คือ mushrooms\_train และ mushrooms\_test และทำการสร้างแบบจำลอง โดยใช้ data จาก mushrooms\_train

ทำการ plot tree ด้วยคำสั่ง rpart.plot

code:

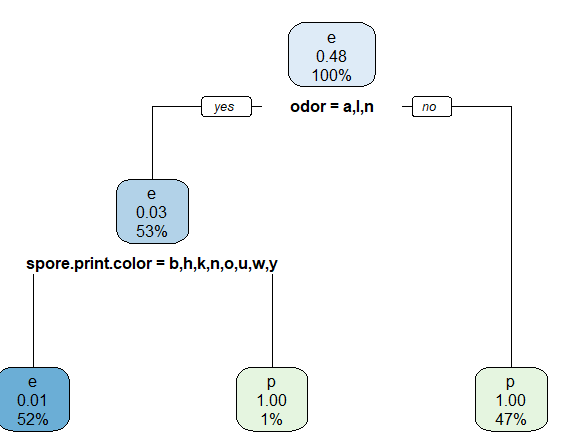
dt.Model <- rpart( mushrooms, data = mushrooms\_train )

dt.Prediction <- predict( dt.Model, newdata = mushrooms\_test,type="class" )

rpart.plot(dt.Model)

test <- table(mushrooms\_test$class,dt.Prediction)

print(confusionMatrix(test))



ผลการทดสอบ สามารถ prediction ได้ความถูกต้อง 99.57%

dt.Prediction

e p

e 857 0

p 7 760

Accuracy : 0.9957

95% CI : (0.9911, 0.9983)

No Information Rate : 0.532

P-Value [Acc > NIR] : < 2e-16

Kappa : 0.9913

Mcnemar's Test P-Value : 0.02334

Sensitivity : 0.9919

Specificity : 1.0000

Pos Pred Value : 1.0000

Neg Pred Value : 0.9909

Prevalence : 0.5320

Detection Rate : 0.5277

Detection Prevalence : 0.5277

Balanced Accuracy : 0.9959

'Positive' Class : e

2.6. สรุปองค์ความรู้ที่ได้จากการใช้แบบจำลองในการแก้ปัญหา และสิ่งที่ได้เรียนรู้เกี่ยวกับกระบวนการในการใช้ข้อมูล แก้ปัญหาจากการบ้านนี้

Ans การสร้างdecision tree สามารถที่จะสร้างได้หลายวิธีทั้งแบบ data.tree package, rpart package แต่ก่อนที่เราจะสร้างแบบจำลองต้องทำการศึกษาข้อมูลก่อนว่าเป็นข้อมูลที่สามารถใช้ได้หรือไม่ เช่น บางข้อมูลมี data เป็น null หรือ NA เราก็ต้องพิจารณาว่าควรนำมาใช้หรือลบทิ้ง หรือdata ที่ให้มีค่าเดียวซ้ำทุกข้อมูล จะนำเข้ามาเป็น Feature ในแบบจำลองหรือไม่ เมื่อวิเคราะห์ว่าดาต้าที่ได้มาสามารถนำมาทำแบบจำลองได้ ก็ต้องทำการแบ่งข้อมูลที่เราจะใช้ Train กับ Test โดยการแบ่งข้อมูลในการTrainควรที่จะมากกว่าข้อมูลในการTest เพราะยิ่งมีข้อมูลที่มีการTrain แบบจำลองก็จะมีประสิทธิภาพสูง ส่วนมากจะแบ่งการTrain = 70 % และ Test = 30% เมื่อทำการแบ่งข้อมูล จากนั้นเราก็ต้องวิเคราะห์ว่า attribute ใดเป็นตัวที่มีความสำคัญในการสร้างแบบจำลอง ซึ่งสามารถหาได้จากการหา Information gain ที่สูงสุดจะเป็นตัวสำคัญในการสร้างแบบจำลอง จากนั้นสร้างตัวแบบจำลองต้นไม้จาก Information gain ของ attribute ที่มีค่าสูงสุด และควรที่จะเช็คความถูกต้องของแบบจำลองโดยการทดสอบข้อมูล ด้วย Confusion Matrix