1) การเตรียมชุดข้อมูล (Data Acquisition)

- อ่านข้อมูลโดยการใช้ฟังก์ชัน read.csv

```
#1.read mushroom data
data <- read.csv(file="/Users/golf/Desktop/R/mushrooms.csv",header = TRUE)
str(data)
print(summary(data))
print(levels(data$odor))
```

- ใช้ฟังก์ชัน str() เพื่อดูโครงสร้างของข้อมูล , ใช้ฟังก์ชัน summary() เพื่อดูว่าแต่ละ attribute มีค่าที่เป็นไปได้คือค่าอะไรบ้าง และ ดูค่าแต่ละค่าว่ามีกี่ instance ที่เป็นค่านั้นๆ , ใช้ฟังก์ชัน levels() เพื่อดูเจาะจงไปแต่ละ attribute ว่ามีค่าที่เป็นไปได้คือค่าอะไรบ้าง เนื่องจากกรณี attrtibute มีค่าที่เป็นไปได้มาก การแสดงผลของฟังก์ชัน summary() จะแสดงผลไม่ครบ (แสดงเป็น Other) , ดูค่าที่เป็นไปได้ (ตัวอักษรย่อ) ว่าคือค่าอะไร จาก https://www.kaggle.com/mokosan/mushroom-classification/data เพื่อทำความเข้าใจข้อมูล

About this file

Attribute Information: (classes: edible=e, poisonous=p)

- cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
- cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s
- cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r,pink=p,purple=u,red=e,white=w,yellow=y
- bruises: bruises=t,no=f
- odor: almond=a,anise=l,creosote=c,fishy=y,foul=f,musty=m,none=n,pungent=p,spicy=s
- gill-attachment: attached=a,descending=d,free=f,notched=n
- gill-spacing: close=c,crowded=w,distant=d
- gill-size: broad=b,narrow=n
- gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g, green=r,orange=o,pink=p,purple=u,red=e,white=w,yellow=y
- stalk-shape: enlarging=e,tapering=t
- stalk-root: bulbous=b,club=c,cup=u,equal=e,rhizomorphs=z,rooted=r,missing=?
- stalk-surface-above-ring: fibrous=f,scaly=y,silky=k,smooth=s
- stalk-surface-below-ring: fibrous=f,scaly=y,silky=k,smooth=s
- stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y
- stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y
- veil-type: partial=p,universal=u
- veil-color: brown=n,orange=o,white=w,yellow=y
- ring-number: none=n,one=o,two=t
- ring-type: cobwebby=c,evanescent=e,flaring=f,large=l,none=n,pendant=p,sheathing=s,zone=z
- spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r,orange=o,purple=u,white=w,yellow=y
- population: abundant=a,clustered=c,numerous=n,scattered=s,several=v,solitary=y
- habitat: grasses=g,leaves=l,meadows=m,paths=p,urban=u,waste=w,woods=d

- เข้าใจข้อมูลว่า ข้อมูลมี type เป็น data.frame ข้อมูลมีทั้งหมด 8124 instances , 23 attributes , attribute แต่ละตัวมี type เป็น factor

```
'data.frame':   8124 obs. of  23 variables:
 $ class                   : Factor w/ 2 levels
 $ cap.shape               : Factor w/ 6 levels
 $ cap.surface             : Factor w/ 4 levels
 $ cap.color               : Factor w/ 10 levels
 $ bruises                 : Factor w/ 2 levels
 $ odor                    : Factor w/ 9 levels
 $ gill.attachment         : Factor w/ 2 levels
 $ gill.spacing            : Factor w/ 2 levels
 $ gill.size               : Factor w/ 2 levels
 $ gill.color              : Factor w/ 12 levels
 $ stalk.shape             : Factor w/ 2 levels
 $ stalk.root              : Factor w/ 5 levels
 $ stalk.surface.above.ring: Factor w/ 4 levels
 $ stalk.surface.below.ring: Factor w/ 4 levels
 $ stalk.color.above.ring  : Factor w/ 9 levels
 $ stalk.color.below.ring  : Factor w/ 9 levels
 $ veil.type               : Factor w/ 1 level "
```

2) การแบ่งข้อมูลเพื่อ Train และ Test แบบจำลอง (Data partitioning)

- แบ่ง train set เป็น 80% และ test set เป็น 20% เพราะการที่ข้อมูลที่ train เยอะ จะทำให้โมเดลมีความแม่นยำมาขึ้น และ fit มากขึ้น

```
#2.split data
index <- 1:nrow(data)
sample_size <- floor(0.8*nrow(data))
set.seed(12345)
testindex <- sample(nrow(data),size=sample_size)

train <- data[testindex,]
test <- data[-testindex,]
```

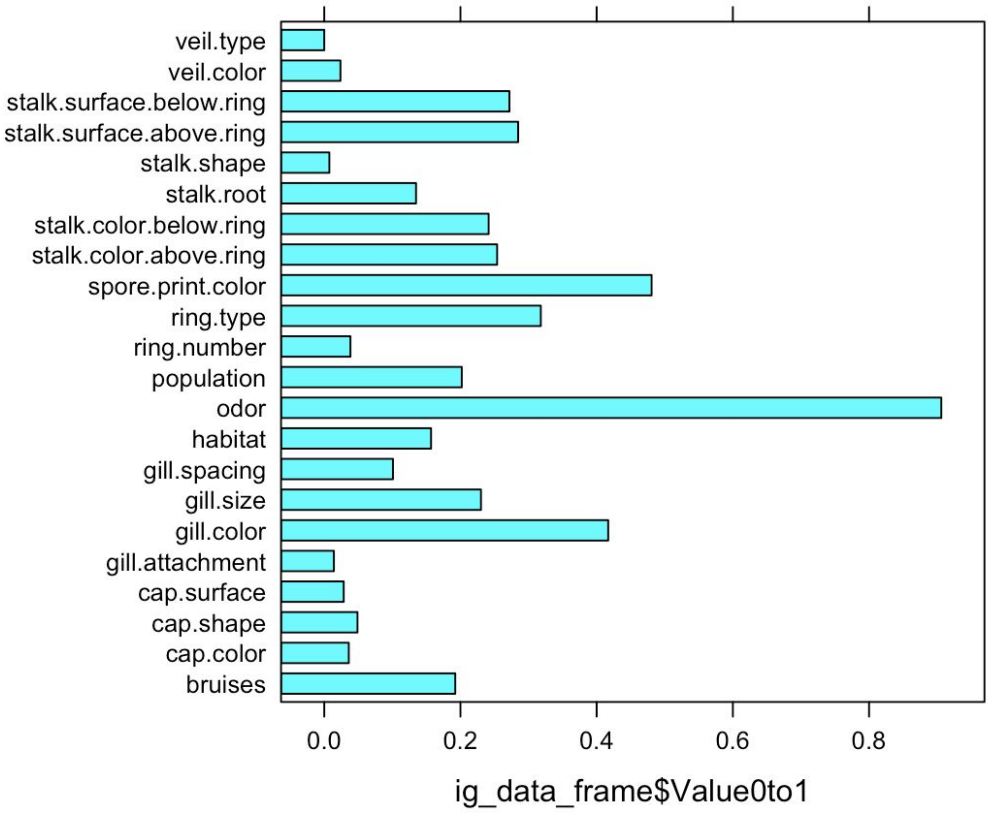
3) การเลือก Attribute เพื่อสร้างแบบจำลอง (Attribute selection)

- ใช้หลักการหา Information Gain เพื่อนำตัวที่มี Information Gain มากที่สุดมาเป็น Root Node เนื่องจากถ้าค่า Inforamtion Gain มาก แสดงว่า Attribute นั้นมีความ Pure มากจึงเหมาะสมกับการจะเป็น Root Node ซึ่งการจะหา Information Gain นั้นต้องใช้ Entropy เพื่อหาความแตกต่างหรือการกระจายของข้อมูล ยิ่ง Entropy มีค่าต่ำ ความแตกต่างของข้อมูลก็จะน้อย

```
> Attribute_Selection(data)
               cap.shape              cap.surface                cap.color                  bruises
            0.048796702             0.028590233             0.036049283             0.192379486
                   odor          gill.attachment            gill.spacing               gill.size
            0.906074977             0.014165027             0.100883184             0.230154375
             gill.color              stalk.shape              stalk.root stalk.surface.above.ring
            0.416977523             0.007516773             0.134817638             0.284725599
stalk.surface.below.ring   stalk.color.above.ring   stalk.color.below.ring                veil.type
            0.271894473             0.253845173             0.241415567             0.000000000
             veil.color              ring.number                ring.type        spore.print.color
            0.023817016             0.038452669             0.318021511             0.480704918
             population                  habitat
            0.201958019             0.156833605
```

4) ค่าที่ได้ จะเห็นว่า Attribute “Odor” มีค่า Information Gain มากที่สุดจึงทำให้ควร Select Attribute นี้มาใช้ใน Root Node

```
#3.attribute selection
Attribute_Selection <- function(data){
  ig <- sapply(colnames(data)[-1], function(x) InformationGain(table(data[,x], data[,1])))
  print(ig)

  #4 visualize
  data_name <- names(ig)
  data_ig <- unname(ig)
  ig_data_frame <- data.frame("Value0to1"= unname(data_ig),"Name"= data_name)
  graph <- barchart(ig_data_frame$Name ~ ig_data_frame$Value0to1, data = ig_data_frame)
  print(graph)
}
```



5.1) การสร้าง Tree จะเริ่มจากการหาความ Pure ของข้อมูล กรณีที่ข้อมูลไม่ Pure จะต้องการหา Information Gain เพื่อหา Feature ที่ทำให้ข้อมูลชุดที่แบ่งมานั้น Pure แล้วจึงนำมาสร้าง Tree ต่อไป

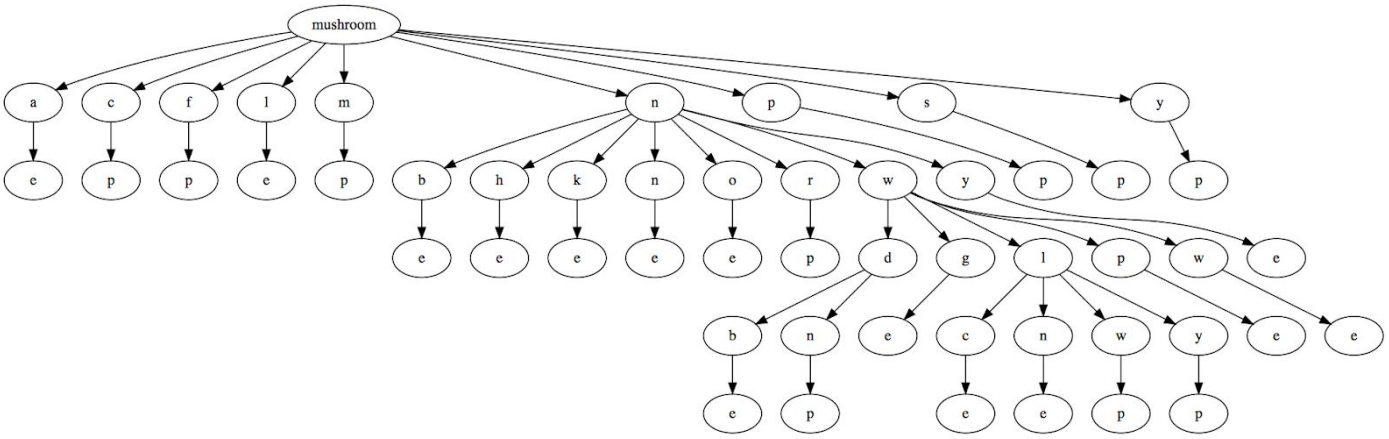
```
#5.1 create tree
IsPure <- function(data){
  length(unique(data[,1])) ==1 #first column = target value (class)
}
Entropy <- function(vls){
  res <- vls/sum(vls) * log2(vls/sum(vls))
  res[vls==0] <- 0
  -sum(res)
}

InformationGain <- function(tble){
  tble <- as.data.frame.matrix(tble)
  entropyBefore <- Entropy(colSums(tble))
  s <- rowSums(tble)
  entropyAfter <- sum (s/sum(s)*apply(tble,MARGIN=1,FUN=Entropy))
  informationGain <- entropyBefore - entropyAfter
  return (informationGain)
}
```

```
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))
}
```

วิธีการสร้างต้นไม้

```
mushroom_tree <- Node$new("mushroom")
TrainID3(mushroom_tree,data)
print(mushroom_tree,"feature","obsCount")
plot(mushroom_tree)
```

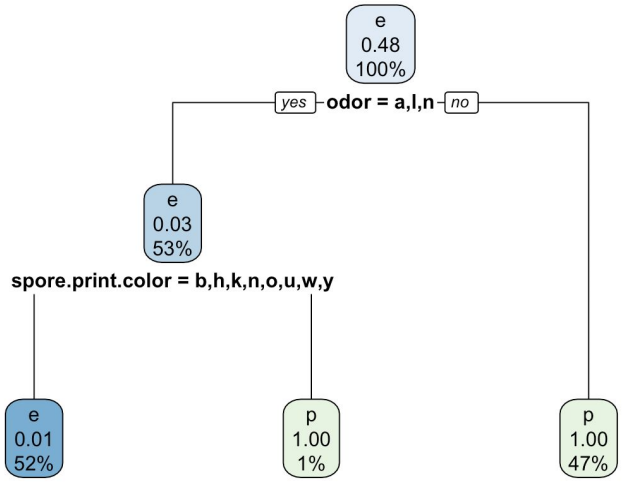
ผลลัพธ์ของการสร้างต้นไม้ ซึ่งตัว leaf node จะเป็น target value



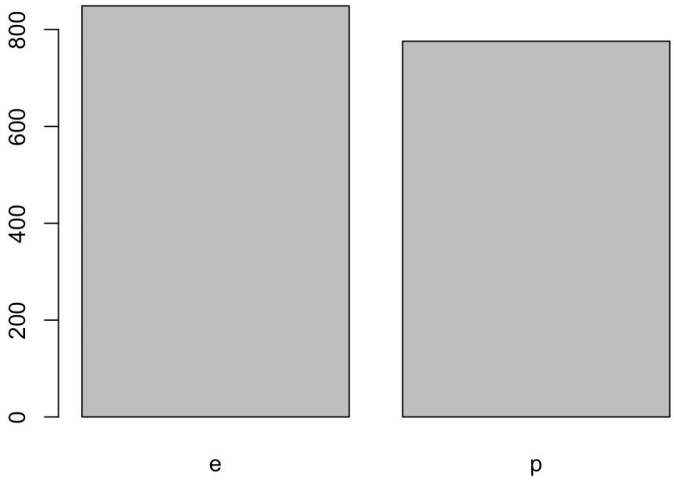
5.2) การสร้าง decision tree โดยใช้ rpart จะใช้ rpart function เพื่อสร้าง model ขึ้นมา โดยจะต้องใส่ parameter formular “class(target value) ~ . (all attributes in data) ” , method = “class” เพื่อให้สร้าง model แบบ classification กรณีไม่ใส่ method ตัว model จะเป็น Regression (method= “anova”) ซึ่งจะให้ค่าที่เป็นตัวเลขออกมา , predict function parameter ตัวแรกจะใส่ model ที่สร้างขึ้นมาจาก rpart function parameter ตัวที่สองจะใส่ test dataset parameter ตัวสุดท้ายจะใส่ type ที่จะใช้ในการ predict ซึ่งต้องใส่ type=”class” เพื่อให้ predict ในรูปแบบของ classification จะได้ข้อมูลหลัง predict แล้วออกมา Label กรณีไม่ใส่ type predict function จะใช้รูปแบบ regression ในการ predict ซึ่งจะให้ค่าออกมาเป็นตัวเลข

```
#5.2 rpart tree
fit <- rpart(class ~ . ,data = train,method = "class")
rpart.plot(fit)
pred_r <- predict(fit, test,type="class")
plot(pred_r)
print(confusionMatrix(table(pred_r,test$class)))
```

ผลลัพธ์จากการสร้าง Decision Tree ด้วย Rpart



ผลลัพธ์จากการ Predict โดยใช้ Model ที่สร้างจาก Rpart



ผลลัพธ์จากการใช้ Confusion Matrix

```
Confusion Matrix and Statistics


pred_r   e   p
     e 840   9
     p   0 776

               Accuracy : 0.9945
                 95% CI : (0.9895, 0.9975)
    No Information Rate : 0.5169
    P-Value [Acc > NIR] : < 2.2e-16


                  Kappa : 0.9889
 Mcnemar's Test P-Value : 0.007661

            Sensitivity : 1.0000
            Specificity : 0.9885
         Pos Pred Value : 0.9894
         Neg Pred Value : 1.0000
             Prevalence : 0.5169
         Detection Rate : 0.5169
   Detection Prevalence : 0.5225
      Balanced Accuracy : 0.9943

       'Positive' Class : e
```

6) สรุปองค์ความรู้ที่ได้จากการใช้แบบจำลองในการแก้ปัญหา และสิ่งที่ได้เรียนรู้เกี่ยวกับกระบวนการในการใช้ข้อมูลแก้ปัญหาจากการบ้านนี้

- ได้รู้การสร้างแบบจำลองโดยใช้วิธีต่างๆ ซึ่งการสร้างแบบจำลองจะมีการสร้างได้หลายวิธี
- ได้รู้การใช้ฟังก์ชันในการสร้างแบบจำลอง รู้ว่าควรใส่ paramater อะไรบ้าง เพื่อที่จะสร้างแบบจำลองนั้นได้
- ได้รู้วิธีการแบ่งชุดข้อมูลให้เป็น train , test data set
- ได้รู้วิธีการศึกษาข้อมูลผ่านการเขียนโค้ด
- ได้ความสามารถในการ Search มากขึ้น