Assignment 4 - The case of Overfitting

Matthew Yu

2017/4

仿照上課講義的 Pima 案例

1. 利用iris資料庫，進行 5-fold 交叉分析。

2. 對每1個fold，計算正確率。

3. 對這5個folds，計算正確率的平均數與標準差。

4. 輸出結果必須包含：每個fold所使用的cp、葉節點個數、正確度。5個folds正確度的平均數與標準差。

針對於這次的作業，我希望能夠讓此流程應用更加廣泛，因此設計一函數。此函數 Generic_cv 的參數有三，分別為 data.frame 型別的資料集、資料集內 Response 的變數名稱字串、以及交叉驗證的折數。輸出結果包含:

- 最適 cp、葉節點數、和正確率的報表
- 正確率的平均表現與標準差
- 個別 run 使用最適 cp 參數所得的分類樹圖數張

1. 函數內容

```
library(rpart)
library(MASS)
data(iris)
Generic_cv <- function(input, response, nfold){
  if(nfold==1){
    print("number of folds should be greater than 1")
  }
  else{
  set.seed(100)
  fold.size <- nrow(input)%/%nfold
  order <- sample(c(1:nrow(input)), nrow(input), replace=F)
  fold.index <- matrix(0, fold.size, nfold)
  Answer.table <- matrix(0,nfold,3)
  colnames(Answer.table) <- c("optimal cp", "nodes", "accuracy")
  for(i in 1:nfold){
    fold.index[,i] <- order[(fold.size*(i-1)+1):(fold.size*i)]
  }
  for(i in 1:nfold){
    train <- input[c(fold.index[,-(nfold-i+1)]),]
    test <- input[c(fold.index[,(nfold-i+1)]),]
    train.tree <- rpart(train[,which(colnames(input)==response)]~.,data=train[,-which(colnames(input)==
    cpt <- printcp(train.tree)
    Answer.table[i,1] <- cpt[which.min(cpt[,4]),1]
    Answer.table[i,2] <- cpt[which.min(cpt[,4]),2]+1
    op.tree <- prune(train.tree, cp=cpt[which.min(cpt[,4]),1])
    data.pre <- predict(op.tree, test, type="class")
    plot(op.tree, margin=0.05, main=paste("Run No.",i))
    text(op.tree, use.n=T, cex=0.75)
    par(ask=T)
```

```
    Answer.table[i,3] <- sum(diag(table(test[,which(colnames(input)==response)], data.pre)))/nrow(test)
  }
  cat("\n","\nresponse = ",response,"\n")
  cat("number of folds = ",nfold,"\n")
  cat("Mean accuracy of the ",nfold,"folds: ",mean(Answer.table[,3]),"\n")
  cat("Standard deviation of the ",nfold,"fold accuracies: ",sd(Answer.table[,3]),"\n")
  cat("\nSummary of the ",nfold,"runs: \n")
  return(Answer.table)
  }
}
```

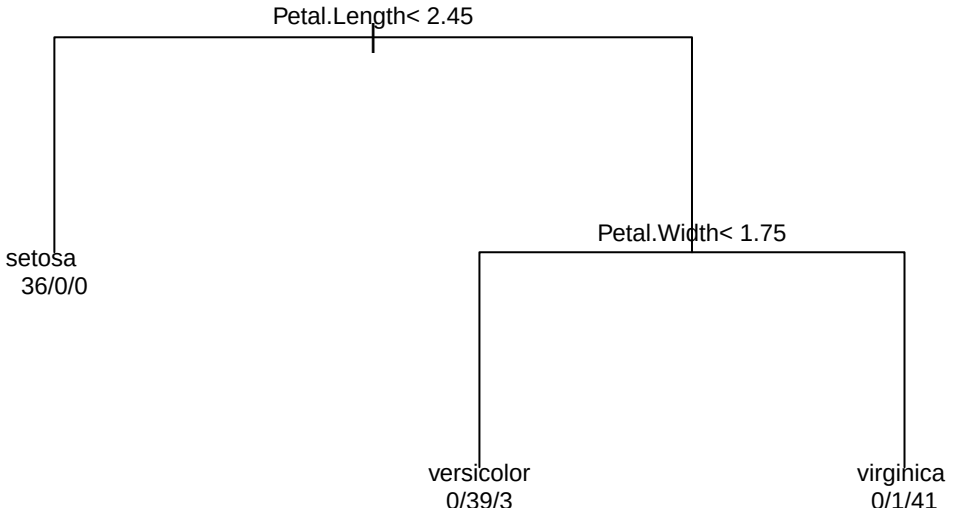
2. 以 Generic_cv(iris, "Species", 5) 為例，可以得結果報表與圖如下:

此外，函數設計上亦能夠適用多種不同資料集與折數

```
Generic_cv(iris, "Species", 5)
```

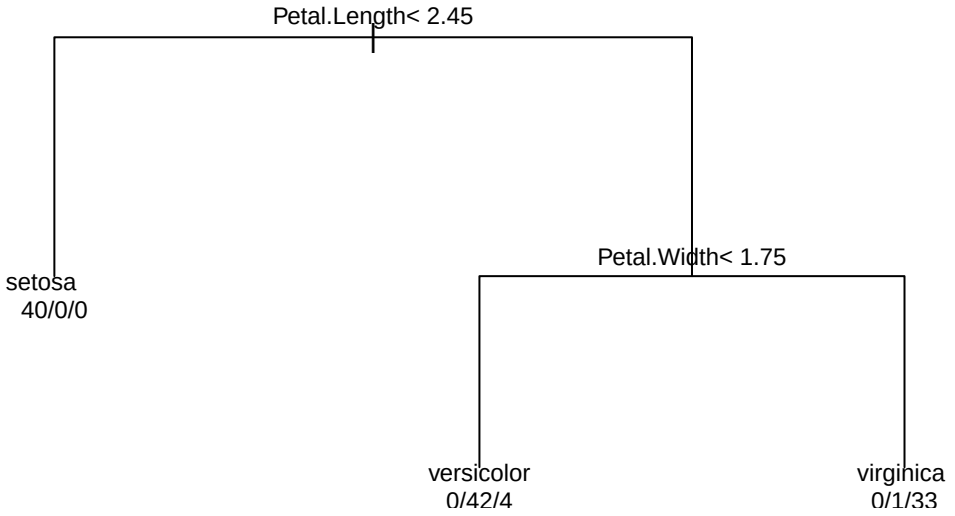
```
##
## Classification tree:
## rpart(formula = train[, which(colnames(input) == response)] ~
##     ., data = train[, -which(colnames(input) == response)])
##
## Variables actually used in tree construction:
## [1] Petal.Length Petal.Width
##
## Root node error: 76/120 = 0.63333
##
## n= 120
##
##        CP nsplit rel error  xerror     xstd
## 1 0.47368      0  1.000000 1.10526 0.066052
## 2 0.01000      2  0.052632 0.11842 0.037965
```

Run No. 1



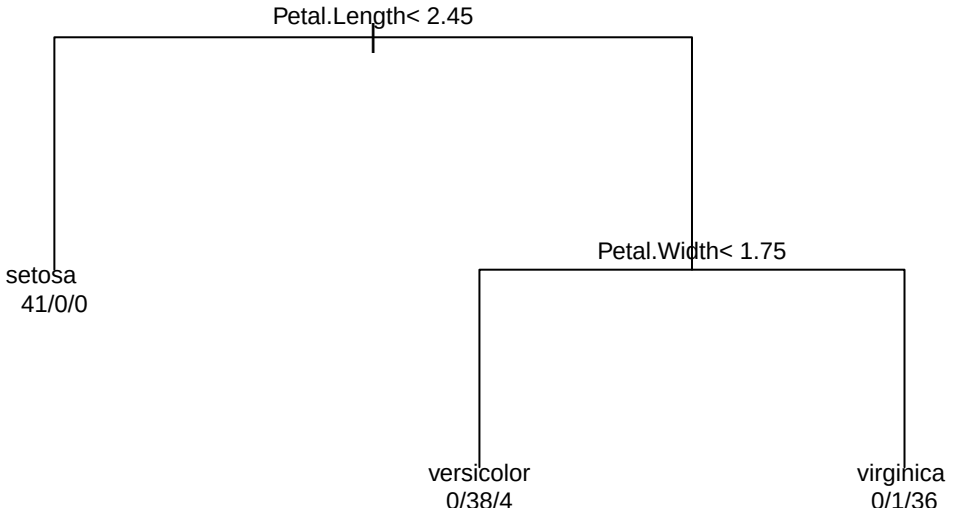

```
##
## Classification tree:
## rpart(formula = train[, which(colnames(input) == response)] ~
##     ., data = train[, -which(colnames(input) == response)])
##
## Variables actually used in tree construction:
## [1] Petal.Length Petal.Width
##
## Root node error: 77/120 = 0.64167
##
## n= 120
##
##        CP nsplit rel error   xerror     xstd
## 1 0.51948      0  1.000000 1.038961 0.067065
## 2 0.41558      1  0.480519 0.480519 0.065699
## 3 0.01000      2  0.064935 0.077922 0.031006
```

Run No. 2



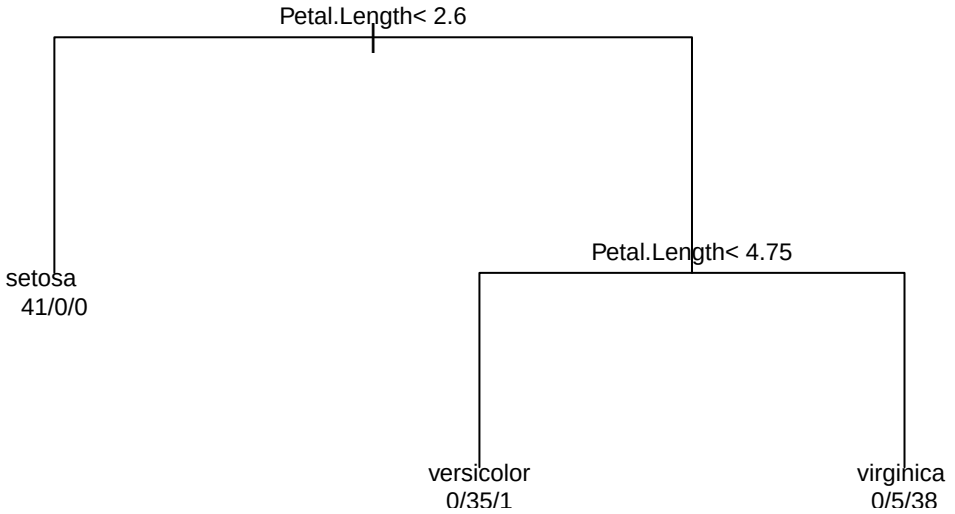

```
##
## Classification tree:
## rpart(formula = train[, which(colnames(input) == response)] ~
##     ., data = train[, -which(colnames(input) == response)])
##
## Variables actually used in tree construction:
## [1] Petal.Length Petal.Width
##
## Root node error: 79/120 = 0.65833
##
## n= 120
##
##        CP nsplit rel error  xerror     xstd
## 1 0.50633      0  1.000000 1.21519 0.055466
## 2 0.43038      1  0.493671 0.68354 0.068984
## 3 0.01000      2  0.063291 0.12658 0.038325
```

Run No. 3



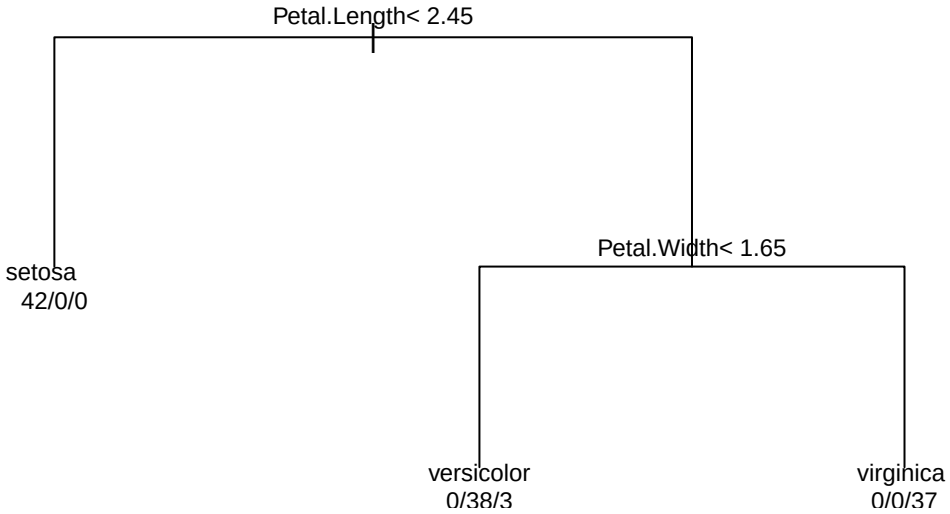

```
##
## Classification tree:
## rpart(formula = train[, which(colnames(input) == response)] ~
##     ., data = train[, -which(colnames(input) == response)])
##
## Variables actually used in tree construction:
## [1] Petal.Length
##
## Root node error: 79/120 = 0.65833
##
## n= 120
##
##        CP nsplit rel error  xerror     xstd
## 1 0.50633      0  1.000000 1.20253 0.056314
## 2 0.41772      1  0.493671 0.62025 0.068157
## 3 0.01000      2  0.075949 0.12658 0.038325
```

Run No. 4




```
##
## Classification tree:
## rpart(formula = train[, which(colnames(input) == response)] ~
##     ., data = train[, -which(colnames(input) == response)])
##
## Variables actually used in tree construction:
## [1] Petal.Length Petal.Width
##
## Root node error: 78/120 = 0.65
##
## n= 120
##
##        CP nsplit rel error   xerror     xstd
## 1 0.51282      0  1.000000 1.192308 0.058646
## 2 0.44872      1  0.487179 0.615385 0.068802
## 3 0.01000      2  0.038462 0.051282 0.025210
```

Run No. 5



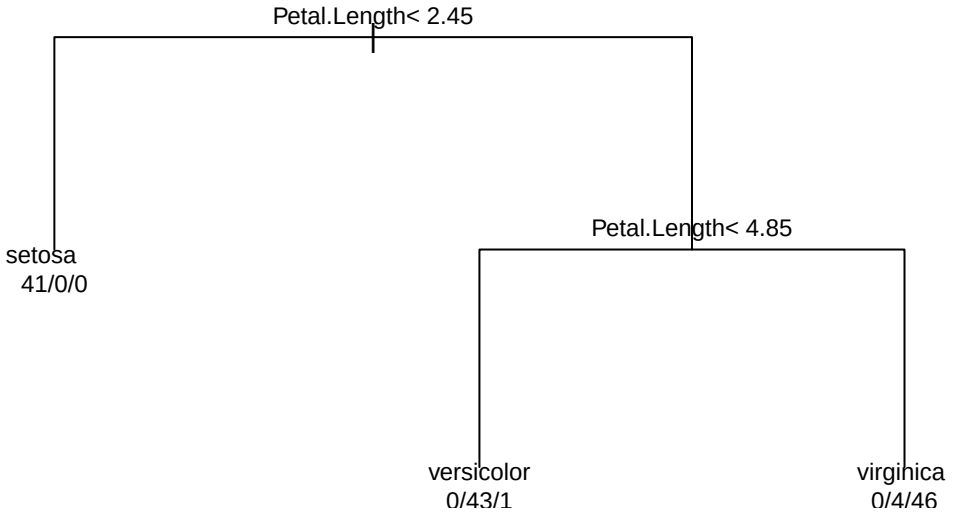

```
##
##
## response =  Species
## number of folds =  5
## Mean accuracy of the  5 folds:  0.9466667
## Standard deviation of the  5 fold accuracies:  0.02981424
##
## Summary of the  5 runs:
##      optimal cp nodes  accuracy
## [1,]       0.01     3 0.9333333
## [2,]       0.01     3 0.9666667
## [3,]       0.01     3 0.9666667
## [4,]       0.01     3 0.9666667
## [5,]       0.01     3 0.9000000
```

```
Generic_cv(iris, "Species", 10)
```

```
##
## Classification tree:
## rpart(formula = train[, which(colnames(input) == response)] ~
##     ., data = train[, -which(colnames(input) == response)])
##
## Variables actually used in tree construction:
## [1] Petal.Length
##
## Root node error: 88/135 = 0.65185
```

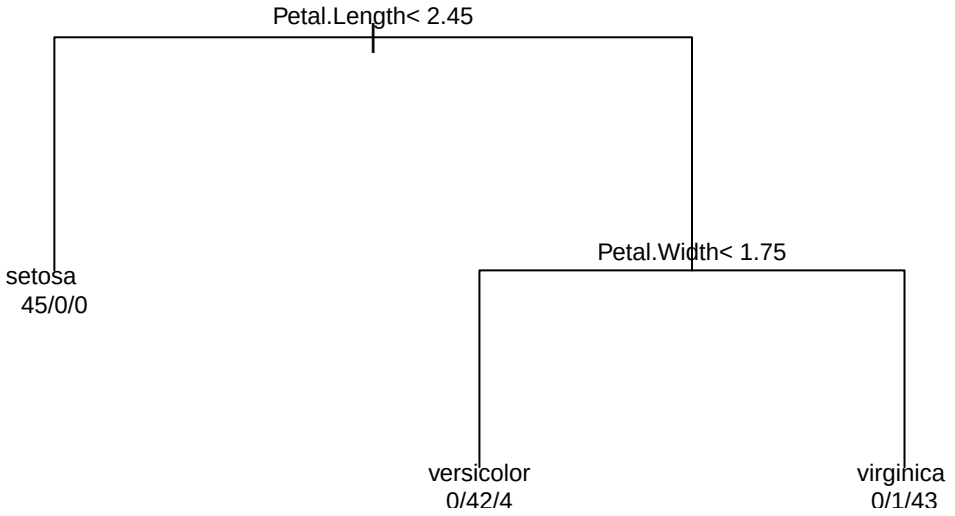
```
##
## n= 135
##
##         CP nsplit rel error  xerror     xstd
## 1 0.47159      0  1.000000 1.25000 0.051288
## 2 0.01000      2  0.056818 0.11364 0.034578
```

Run No. 1



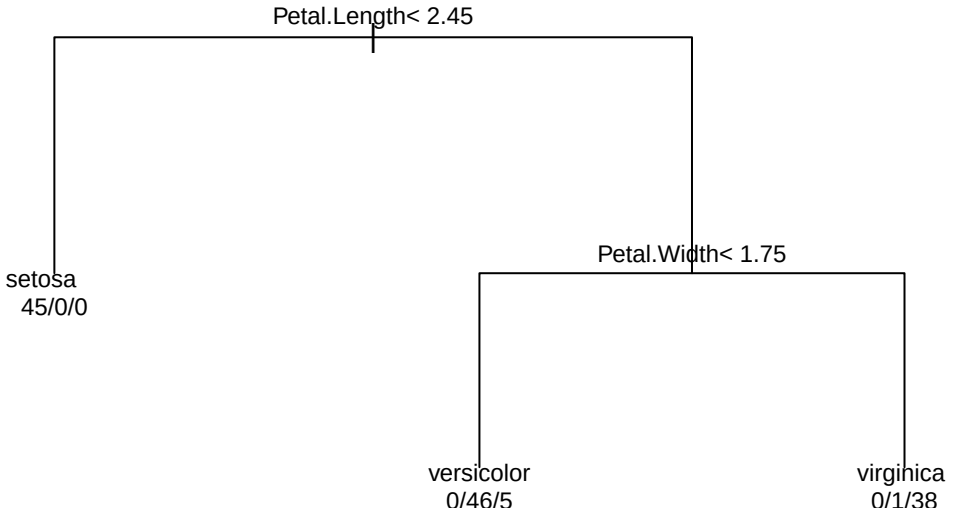

```
##
## Classification tree:
## rpart(formula = train[, which(colnames(input) == response)] ~
##     ., data = train[, -which(colnames(input) == response)])
##
## Variables actually used in tree construction:
## [1] Petal.Length Petal.Width
##
## Root node error: 88/135 = 0.65185
##
## n= 135
##
##         CP nsplit rel error   xerror     xstd
## 1 0.51136      0  1.000000 1.295455 0.047853
## 2 0.43182      1  0.488636 0.681818 0.065608
## 3 0.01000      2  0.056818 0.090909 0.031174
```

Run No. 2



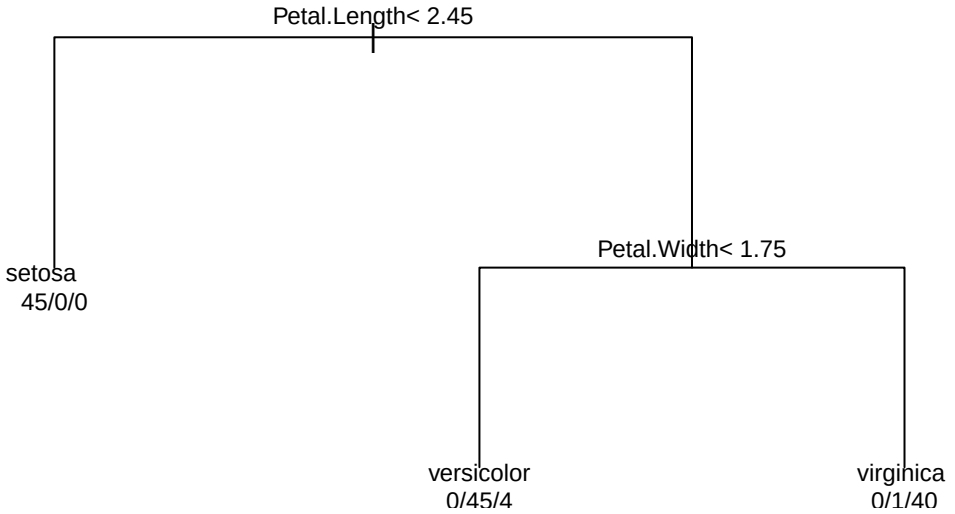

```
##
## Classification tree:
## rpart(formula = train[, which(colnames(input) == response)] ~
##     ., data = train[, -which(colnames(input) == response)])
##
## Variables actually used in tree construction:
## [1] Petal.Length Petal.Width
##
## Root node error: 88/135 = 0.65185
##
## n= 135
##
##        CP nsplit rel error  xerror     xstd
## 1 0.51136      0  1.000000 1.14773 0.057313
## 2 0.42045      1  0.488636 0.54545 0.063202
## 3 0.01000      2  0.068182 0.13636 0.037575
```

Run No. 3



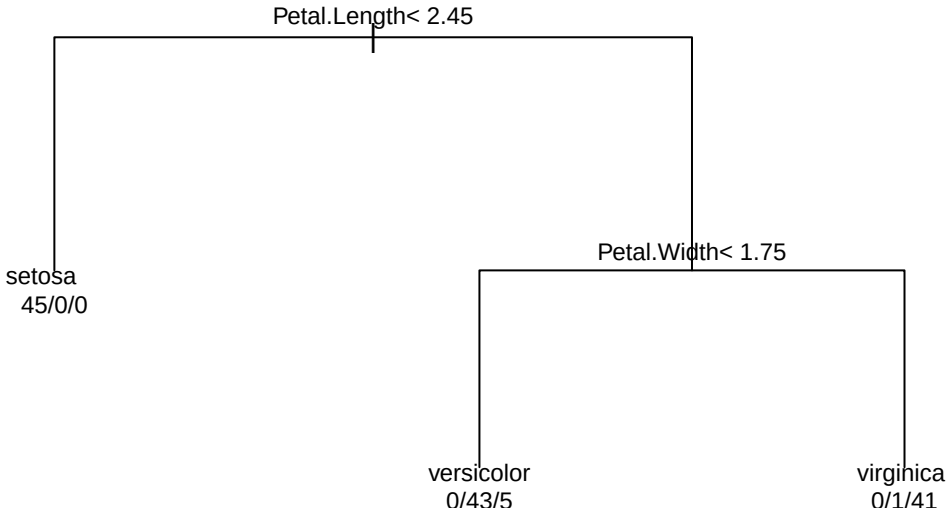

```
##
## Classification tree:
## rpart(formula = train[, which(colnames(input) == response)] ~
##     ., data = train[, -which(colnames(input) == response)])
##
## Variables actually used in tree construction:
## [1] Petal.Length Petal.Width
##
## Root node error: 89/135 = 0.65926
##
## n= 135
##
##        CP nsplit rel error   xerror     xstd
## 1 0.50562      0   1.00000 1.112360 0.057731
## 2 0.43820      1   0.49438 0.528090 0.062192
## 3 0.01000      2   0.05618 0.067416 0.026904
```

Run No. 4



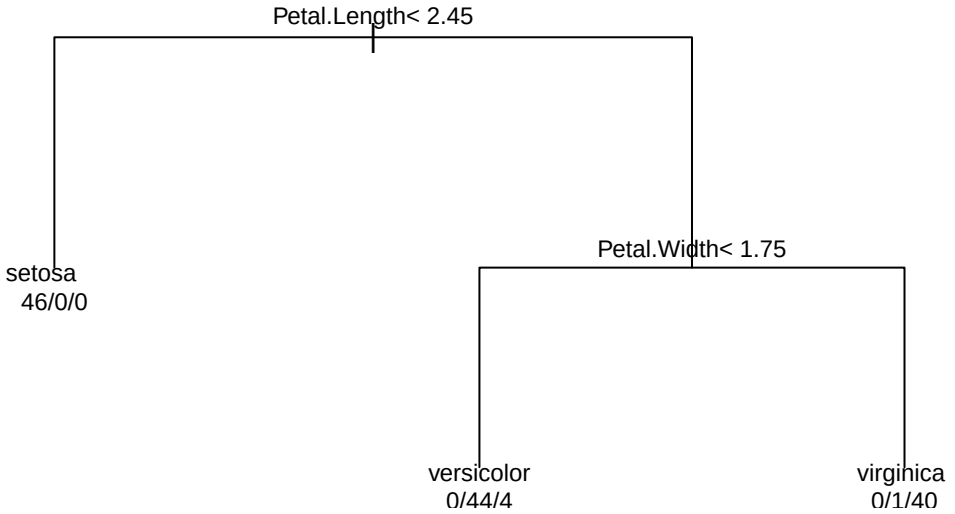

```
##
## Classification tree:
## rpart(formula = train[, which(colnames(input) == response)] ~
##     ., data = train[, -which(colnames(input) == response)])
##
## Variables actually used in tree construction:
## [1] Petal.Length Petal.Width
##
## Root node error: 89/135 = 0.65926
##
## n= 135
##
##        CP nsplit rel error  xerror     xstd
## 1 0.50562      0  1.000000 1.17978 0.054275
## 2 0.42697      1  0.494382 0.67416 0.064871
## 3 0.01000      2  0.067416 0.10112 0.032565
```

Run No. 5



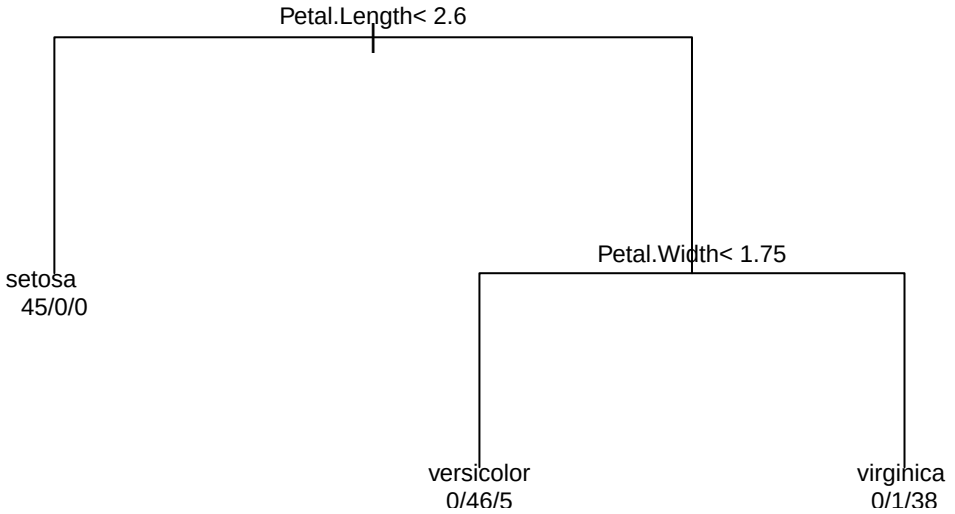

```
##
## Classification tree:
## rpart(formula = train[, which(colnames(input) == response)] ~
##     ., data = train[, -which(colnames(input) == response)])
##
## Variables actually used in tree construction:
## [1] Petal.Length Petal.Width
##
## Root node error: 89/135 = 0.65926
##
## n= 135
##
##        CP nsplit rel error  xerror     xstd
## 1 0.50562      0   1.00000 1.08989 0.058711
## 2 0.43820      1   0.49438 0.57303 0.063295
## 3 0.01000      2   0.05618 0.11236 0.034190
```

Run No. 6



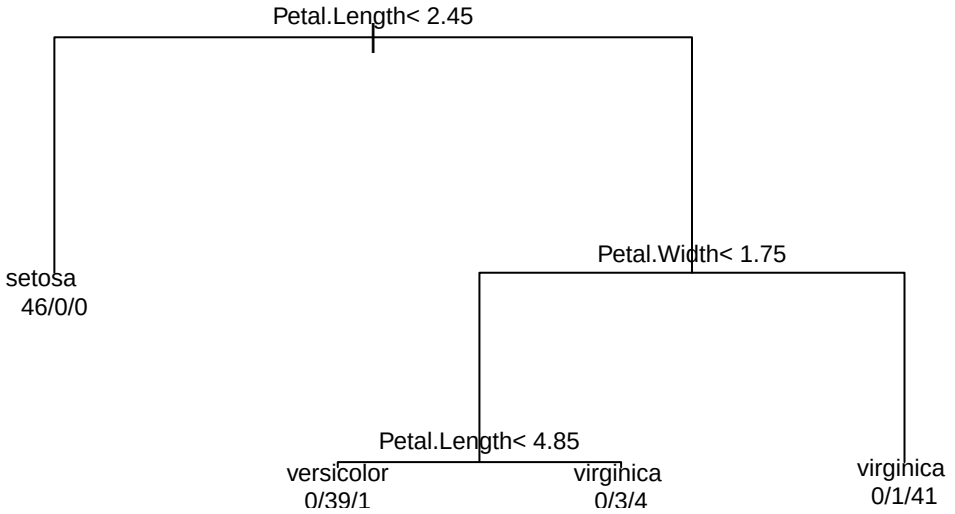

```
##
## Classification tree:
## rpart(formula = train[, which(colnames(input) == response)] ~
##     ., data = train[, -which(colnames(input) == response)])
##
## Variables actually used in tree construction:
## [1] Petal.Length Petal.Width
##
## Root node error: 88/135 = 0.65185
##
## n= 135
##
##        CP nsplit rel error  xerror     xstd
## 1 0.51136      0  1.000000 1.18182 0.055533
## 2 0.42045      1  0.488636 0.55682 0.063489
## 3 0.01000      2  0.068182 0.10227 0.032935
```

Run No. 7



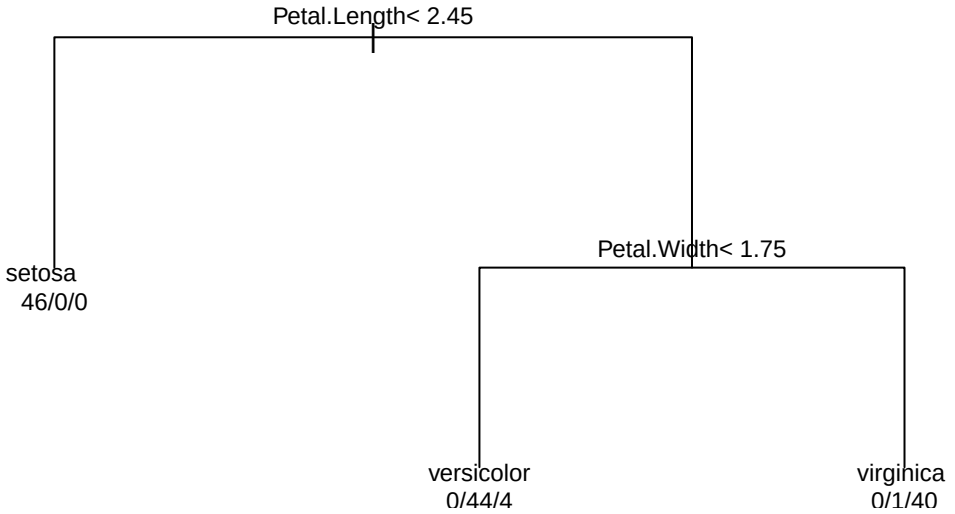

```
##
## Classification tree:
## rpart(formula = train[, which(colnames(input) == response)] ~
##     ., data = train[, -which(colnames(input) == response)])
##
## Variables actually used in tree construction:
## [1] Petal.Length Petal.Width
##
## Root node error: 89/135 = 0.65926
##
## n= 135
##
##         CP nsplit rel error  xerror     xstd
## 1 0.516854      0  1.000000 1.21348 0.052220
## 2 0.415730      1  0.483146 0.76404 0.065273
## 3 0.011236      2  0.067416 0.14607 0.038512
## 4 0.010000      3  0.056180 0.11236 0.034190
```

Run No. 8



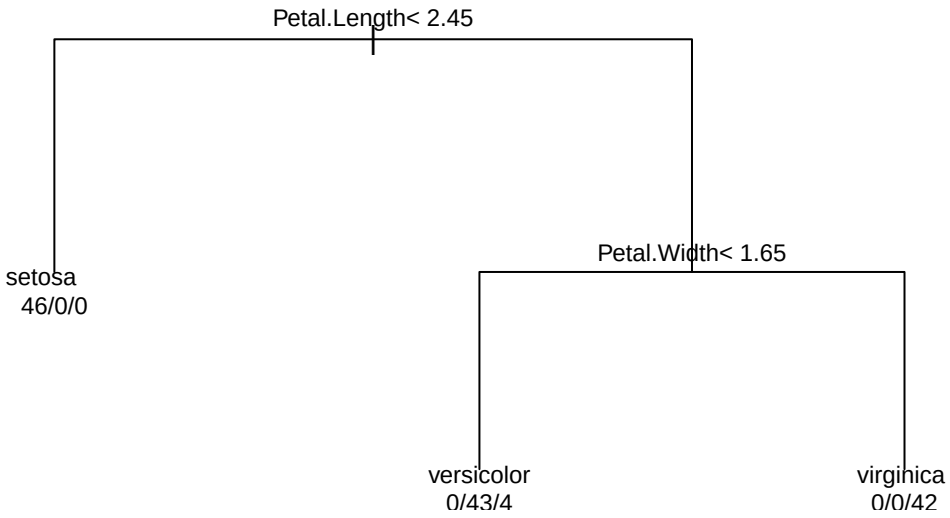

```
##
## Classification tree:
## rpart(formula = train[, which(colnames(input) == response)] ~
##     ., data = train[, -which(colnames(input) == response)])
##
## Variables actually used in tree construction:
## [1] Petal.Length Petal.Width
##
## Root node error: 89/135 = 0.65926
##
## n= 135
##
##        CP nsplit rel error   xerror     xstd
## 1 0.50562      0   1.00000 1.101124 0.058231
## 2 0.43820      1   0.49438 0.584270 0.063531
## 3 0.01000      2   0.05618 0.067416 0.026904
```

Run No. 9




```
##
## Classification tree:
## rpart(formula = train[, which(colnames(input) == response)] ~
##     ., data = train[, -which(colnames(input) == response)])
##
## Variables actually used in tree construction:
## [1] Petal.Length Petal.Width
##
## Root node error: 89/135 = 0.65926
##
## n= 135
##
##         CP nsplit rel error   xerror     xstd
## 1 0.516854      0  1.000000 1.168539 0.054909
## 2 0.438202      1  0.483146 0.550562 0.062776
## 3 0.011236      2  0.044944 0.078652 0.028947
## 4 0.010000      3  0.033708 0.078652 0.028947
```

Run No. 10




```
##
##
## response =  Species
## number of folds =  10
## Mean accuracy of the  10 folds:  0.94
## Standard deviation of the  10 fold accuracies:  0.04919099
##
## Summary of the  10 runs:
```

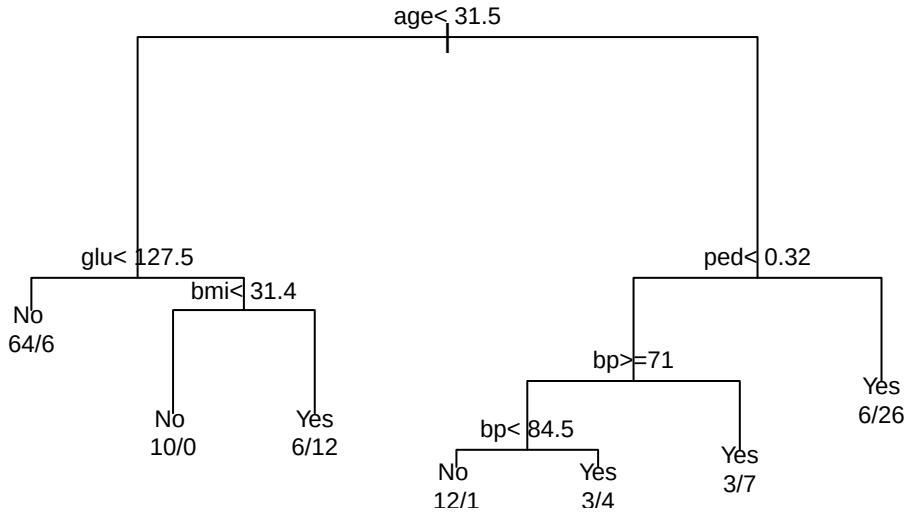
```
##       optimal cp nodes  accuracy
##  [1,] 0.01000000     3 0.8666667
##  [2,] 0.01000000     3 0.9333333
##  [3,] 0.01000000     3 1.0000000
##  [4,] 0.01000000     3 0.9333333
##  [5,] 0.01000000     3 1.0000000
##  [6,] 0.01000000     3 0.9333333
##  [7,] 0.01000000     3 1.0000000
##  [8,] 0.01000000     4 0.9333333
##  [9,] 0.01000000     3 0.9333333
## [10,] 0.01123596     3 0.8666667
```

```
Generic_cv(Pima.tr, "type", 5)
```

```
##
## Classification tree:
## rpart(formula = train[, which(colnames(input) == response)] ~
##     ., data = train[, -which(colnames(input) == response)])
```

```
##
## Variables actually used in tree construction:
## [1] age bmi bp  glu ped
##
## Root node error: 56/160 = 0.35
##
## n= 160
##
##         CP nsplit rel error  xerror    xstd
## 1 0.250000      0   1.00000 1.00000 0.10774
## 2 0.107143      1   0.75000 1.10714 0.11004
## 3 0.071429      2   0.64286 1.05357 0.10898
## 4 0.053571      3   0.57143 0.94643 0.10631
## 5 0.017857      5   0.46429 0.89286 0.10470
## 6 0.010000      6   0.44643 0.80357 0.10156
```

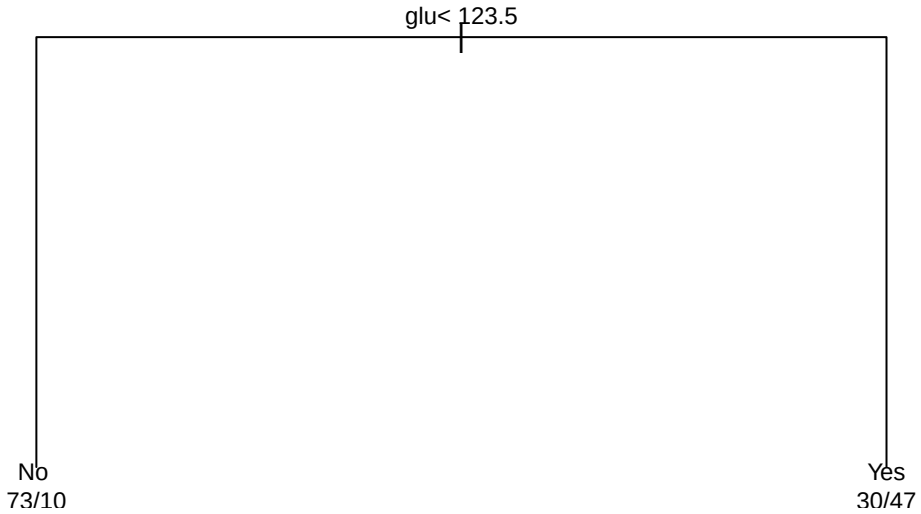
Run No. 1




```
##
## Classification tree:
## rpart(formula = train[, which(colnames(input) == response)] ~
##     ., data = train[, -which(colnames(input) == response)])
##
## Variables actually used in tree construction:
## [1] bmi glu ped
##
## Root node error: 57/160 = 0.35625
##
## n= 160
```

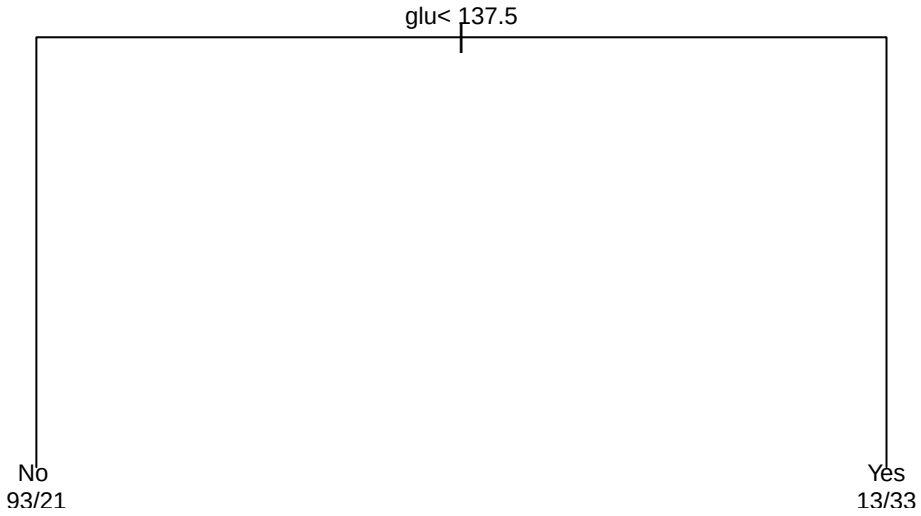
```
##
##         CP nsplit rel error  xerror     xstd
## 1 0.298246      0   1.00000 1.00000 0.106273
## 2 0.140351      1   0.70175 0.71930 0.096879
## 3 0.040936      2   0.56140 0.77193 0.099088
## 4 0.010000      5   0.43860 0.87719 0.102860
```

Run No. 2



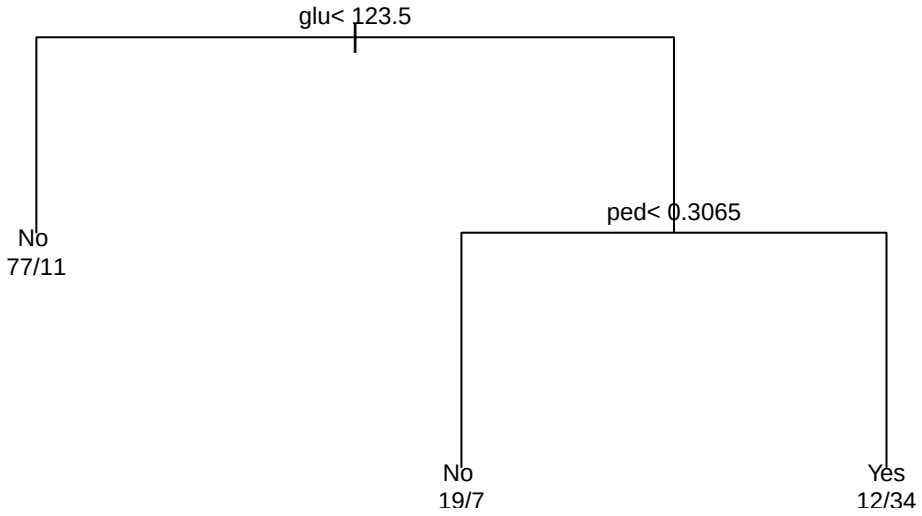

```
##
## Classification tree:
## rpart(formula = train[, which(colnames(input) == response)] ~
##     ., data = train[, -which(colnames(input) == response)])
##
## Variables actually used in tree construction:
## [1] age  bmi  glu  ped  skin
##
## Root node error: 54/160 = 0.3375
##
## n= 160
##
##         CP nsplit rel error  xerror    xstd
## 1 0.370370      0   1.00000 1.00000 0.11076
## 2 0.055556      1   0.62963 0.79630 0.10384
## 3 0.030864      3   0.51852 0.85185 0.10602
## 4 0.010000      8   0.33333 0.98148 0.11025
```

Run No. 3



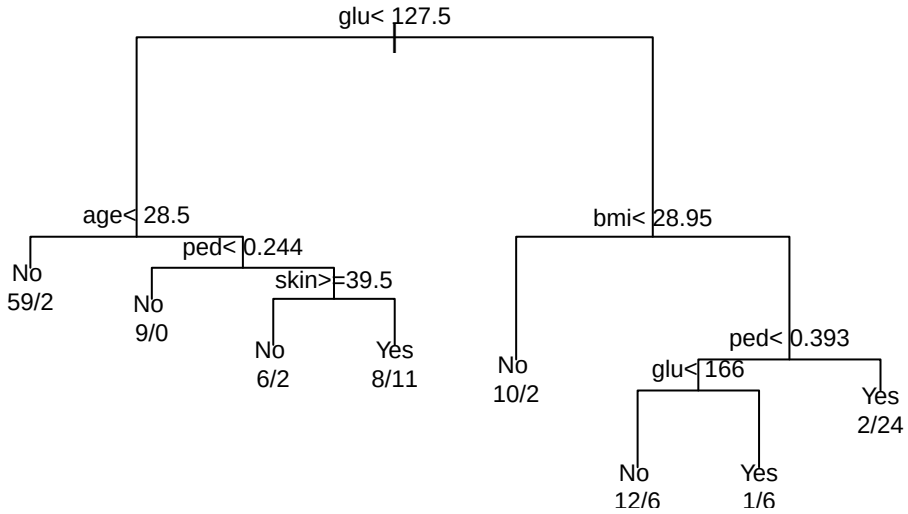

```
##
## Classification tree:
## rpart(formula = train[, which(colnames(input) == response)] ~
##     ., data = train[, -which(colnames(input) == response)])
##
## Variables actually used in tree construction:
## [1] bmi glu ped
##
## Root node error: 52/160 = 0.325
##
## n= 160
##
##         CP nsplit rel error  xerror     xstd
## 1 0.211538      0   1.00000 1.00000 0.113933
## 2 0.057692      2   0.57692 0.63462 0.098423
## 3 0.010000      3   0.51923 0.76923 0.105331
```

Run No. 4




```
##
## Classification tree:
## rpart(formula = train[, which(colnames(input) == response)] ~
##     ., data = train[, -which(colnames(input) == response)])
##
## Variables actually used in tree construction:
## [1] age  bmi  glu  ped  skin
##
## Root node error: 53/160 = 0.33125
##
## n= 160
##
##         CP nsplit rel error  xerror    xstd
## 1 0.245283      0   1.00000 1.00000 0.11233
## 2 0.150943      1   0.75472 1.15094 0.11592
## 3 0.056604      2   0.60377 0.86792 0.10802
## 4 0.018868      4   0.49057 0.77358 0.10419
## 5 0.010000      7   0.43396 0.75472 0.10334
```

Run No. 5




```
##
##
## response =  type
## number of folds =  5
## Mean accuracy of the  5 folds:  0.675
## Standard deviation of the  5 fold accuracies:  0.0728869
##
## Summary of the  5 runs:
##      optimal cp nodes accuracy
## [1,] 0.01000000     7    0.725
## [2,] 0.14035088     2    0.675
## [3,] 0.05555556     2    0.550
## [4,] 0.05769231     3    0.700
## [5,] 0.01000000     8    0.725
```