Tree Algorithm

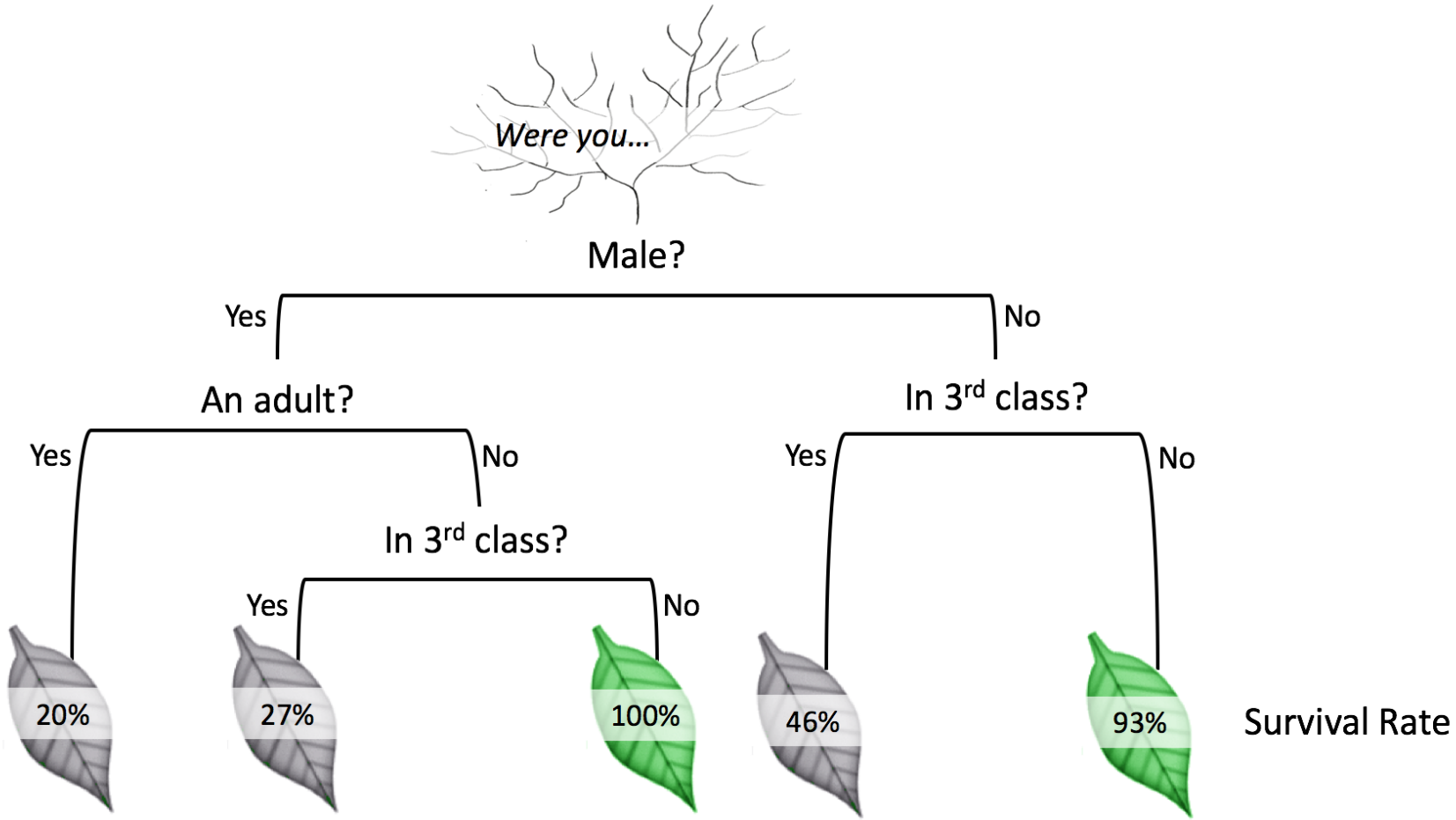
Group 3

Lee Dongkyu & Han Hyemin

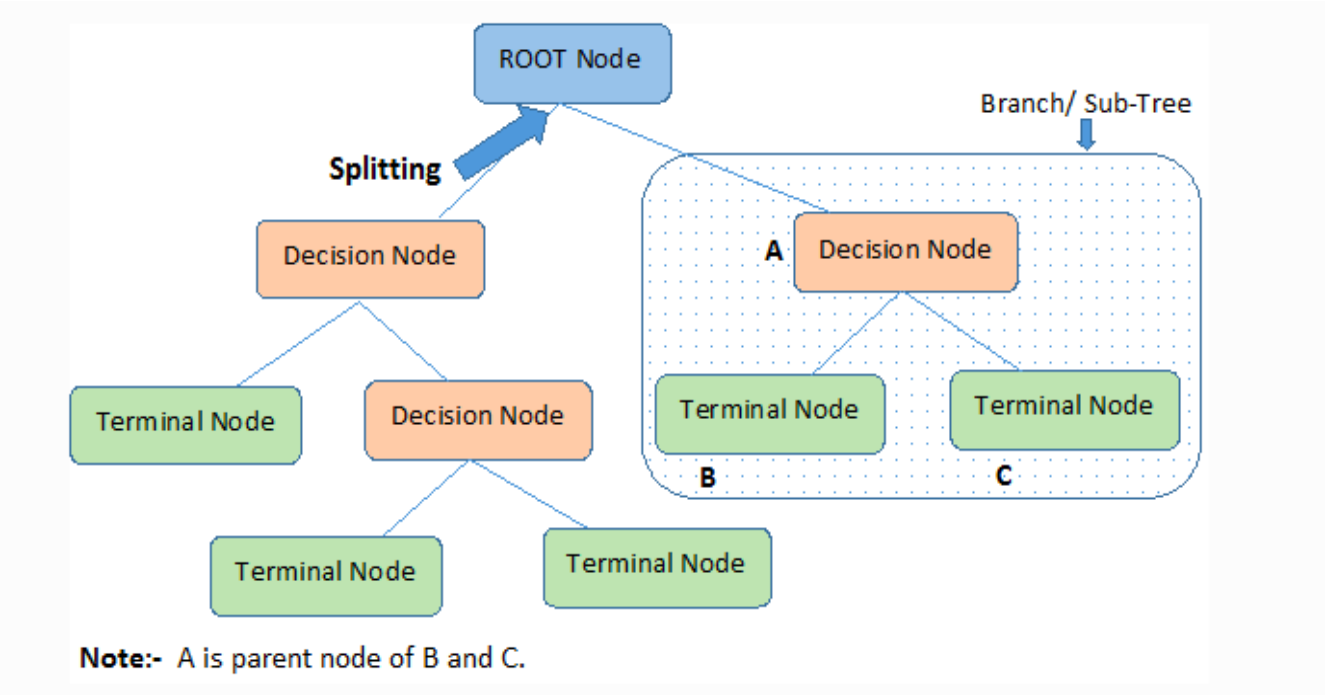
Decision Tree

- Decision Tree belongs to the family of supervised learning and can be used for solving classification and regression problems.($\rightarrow$ CART)

Simple Example - Titanic Case



Notation



- **Root Node** : It represents the entire population or sample.
- **Splitting** : It is a process of dividing a node into two or more sub-nodes.
- **Decision Node** : A sub-node splits into further sub-nodes.
- **Leaf/Terminal Node** : Nodes do not split.
- **Branch/Sub-Tree** : A subsection of the entire tree.

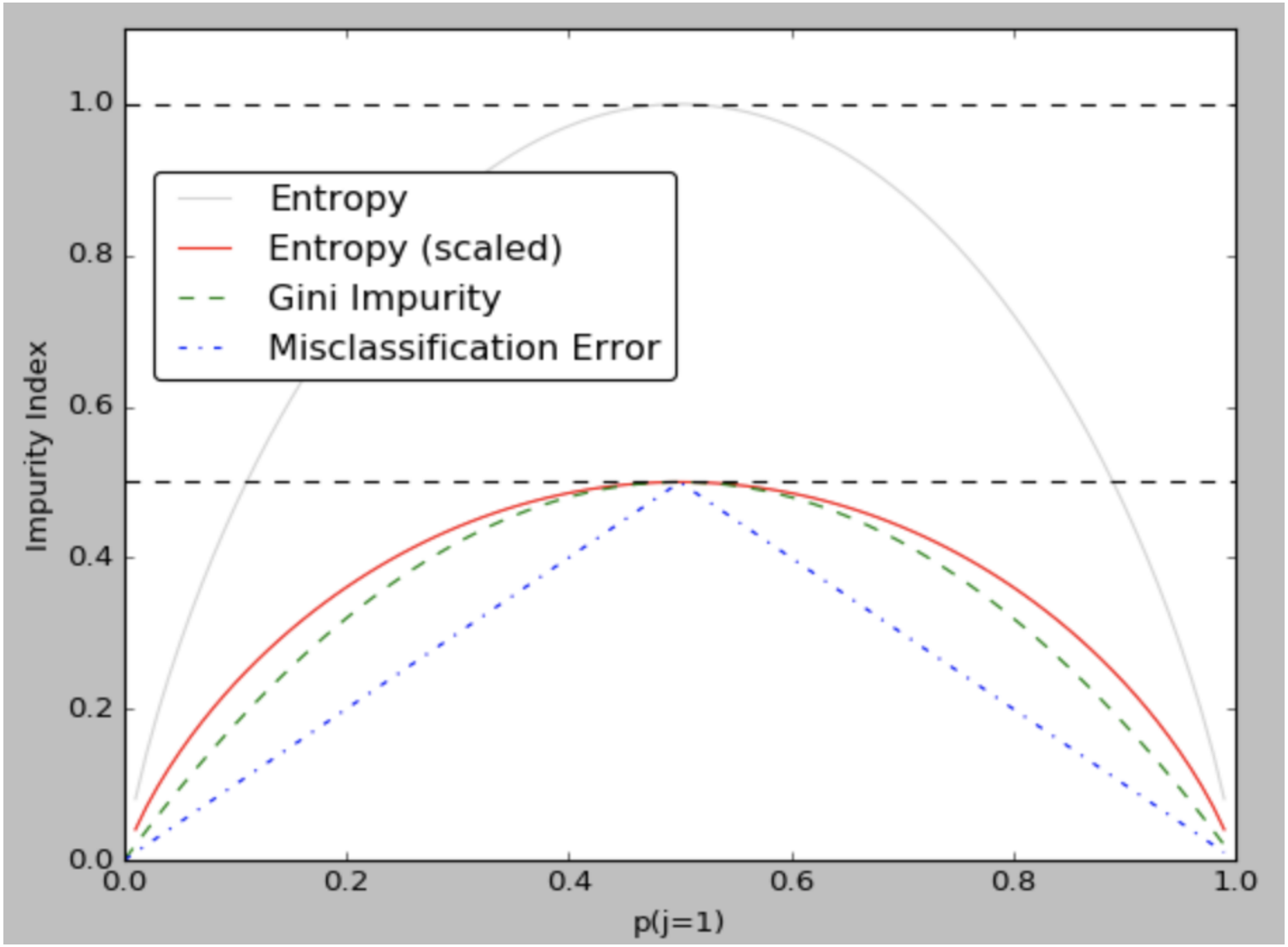
Splitting

- Classification : Minimizing Impurity
  - Gini index : $\sum_{j=1}^{J} \hat{p}_j(1-\hat{p}_j)$
    - J : number of classes
    - $\hat{p}_j$: estimated probability(ratio) of j^{th}-class
  - Entropy(Information index) : $-\sum_{j=1}^{J} \hat{p}_j \log_2(\hat{p}_j)$
    - J : number of classes
    - $\hat{p}_j$: estimated probability(ratio) of j^{th}-class
  - Misclassification Error : $1 - \max[\hat{p}_1, \cdots, \hat{p}_J]$
    - $\hat{p}_j$: estimated probability(ratio) of j^{th}-class
- Regression : Minimizing RSS (Residual Sum of Squares)

Simple Example - Impurity Calculation

- **Root Node** : Class 1 : 50 / Class 2 : 50
- **Leaf Node 1** : Class 1 : 10 / Class 2 : 40
- **Leaf Node 2** : Class 1 : 40 / Class 2 : 10
- Root Node's Impurity
  - Gini index : $\sum_{j=1}^{J} \hat{p}_j(1-\hat{p}_j) = 0.5 \times 0.5 + 0.5 \times 0.5 = 0.5$
  - Entropy : $-\sum_{j=1}^{J} \hat{p}_j(\log_2 \hat{p}_j) = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$
  - Misclassification Error : $1 - \max[\hat{p_1}, \cdots, \hat{p_J}] = 1 - \max[0.5, 0.5] = 0.5$
- Leaf Node's Impurity (Average)
  - Gini index : $\frac{1}{2}(0.2 \times 0.8 + 0.8 \times 0.2) + \frac{1}{2}(0.8 \times 0.2 + 0.2 \times 0.8) = 0.32$
  - Entropy : $\frac{1}{2}(-0.2 \log_2 0.2 - 0.8 \log_2 0.8) + \frac{1}{2}(-0.8 \log_2 0.8 - 0.2 \log_2 0.2) = 0.72$
  - Misclassification Error : $\frac{1}{2}(1 - \max[0.2, 0.8]) + \frac{1}{2}(1 - \max[0.8, 0.2]) = 0.2$

Gini vs Entropy vs Misclassification (2-Classes Case)



Stop Splitting

- Method#1 : Set maximum depth.(Number of splitting)
- Method#2 : Set criteria of impurity.
  - If decreasing impurity $\leq$ criteria, stop splitting.
- Method#3 : Set number of observation at leaf node.
- Pruning is a technique that reduces the size of decision trees by removing sections of the tree that provide little power to classify instances.
  - Example : Split until observation is 1 at all leaf node by Gini index. After, remove trees what provide little power to classify based on Misclassification error.

Classification Example - Kyphosis Case

```
library(rpart) # package for tree
data(kyphosis)
str(kyphosis)
```

```
## 'data.frame':    81 obs. of  4 variables:
##  $ Kyphosis: Factor w/ 2 levels "absent","present": 1 1 2 1 1 1 1 1 1 2 ...
##  $ Age     : int  71 158 128 2 1 1 61 37 113 59 ...
##  $ Number  : int  3 3 4 5 4 2 2 3 2 6 ...
##  $ Start   : int  5 14 5 1 15 16 17 16 16 12 ...
```

- Kyphosis : A factor with levels absent present indicating if a kyphosis (a type of deformation) was present after the operation.
- Age : In months.
- Numbers : The number of vertebrae involved.
- Start : The number of the first vertebra operated on.

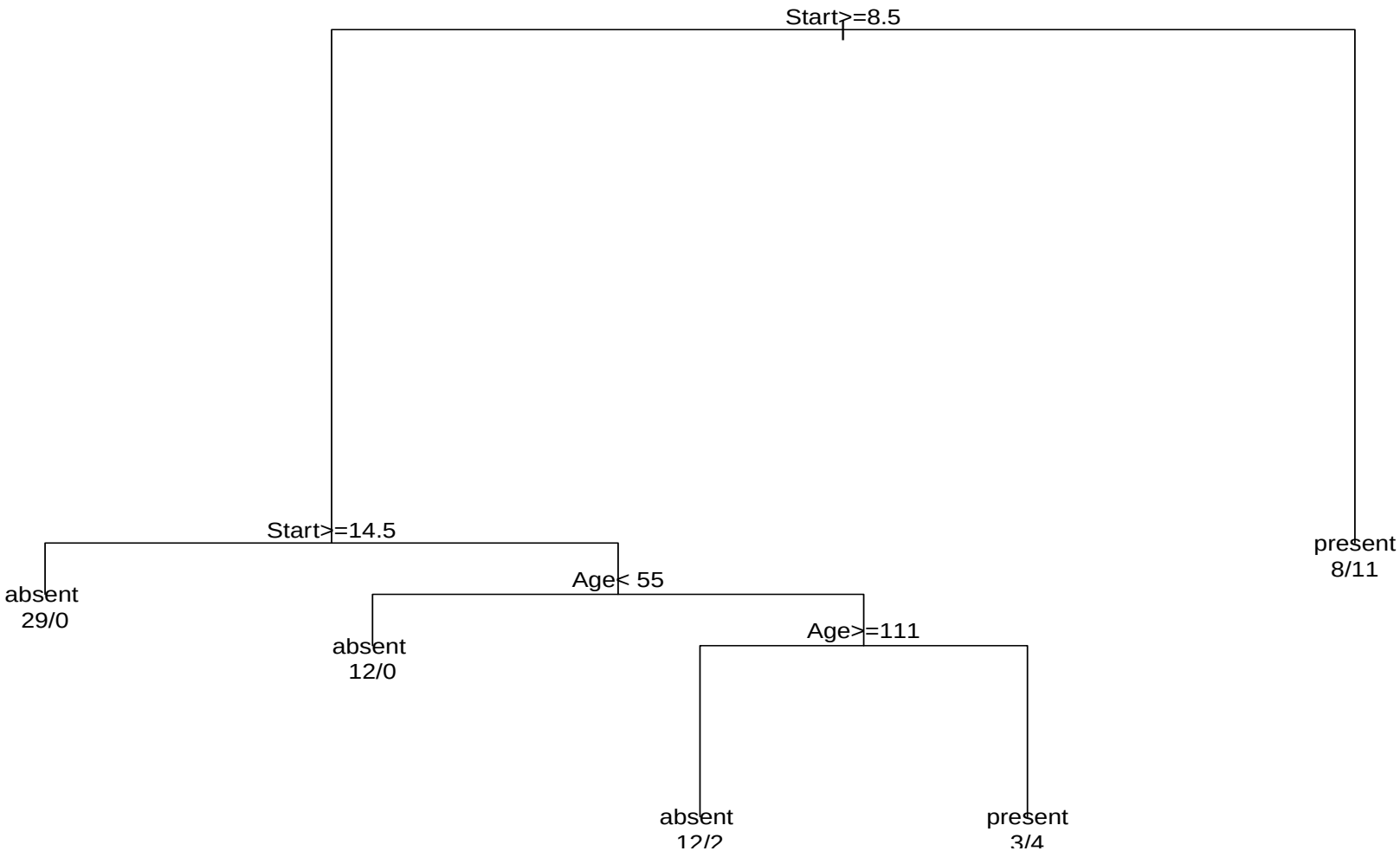
Classification Example - Kyphosis Case (count.)

```
tree_g <- rpart(Kyphosis ~ Age + Number + Start, data=kyphosis,
                parm=list(split="gini"))
tree_g
```

```
## n= 81
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 81 17 absent (0.79012346 0.20987654)
##    2) Start>=8.5 62  6 absent (0.90322581 0.09677419)
##      4) Start>=14.5 29  0 absent (1.00000000 0.00000000) *
##      5) Start< 14.5 33  6 absent (0.81818182 0.18181818)
##       10) Age< 55 12  0 absent (1.00000000 0.00000000) *
##       11) Age>=55 21  6 absent (0.71428571 0.28571429)
##         22) Age>=111 14  2 absent (0.85714286 0.14285714) *
##         23) Age< 111 7  3 present (0.42857143 0.57142857) *
##    3) Start< 8.5 19  8 present (0.42105263 0.57894737) *
```

Classification Example - Kyphosis Case (count.)

```
par(mar=c(1,1,1,1)); plot(tree_g); text(tree_g, use.n=TRUE)
```



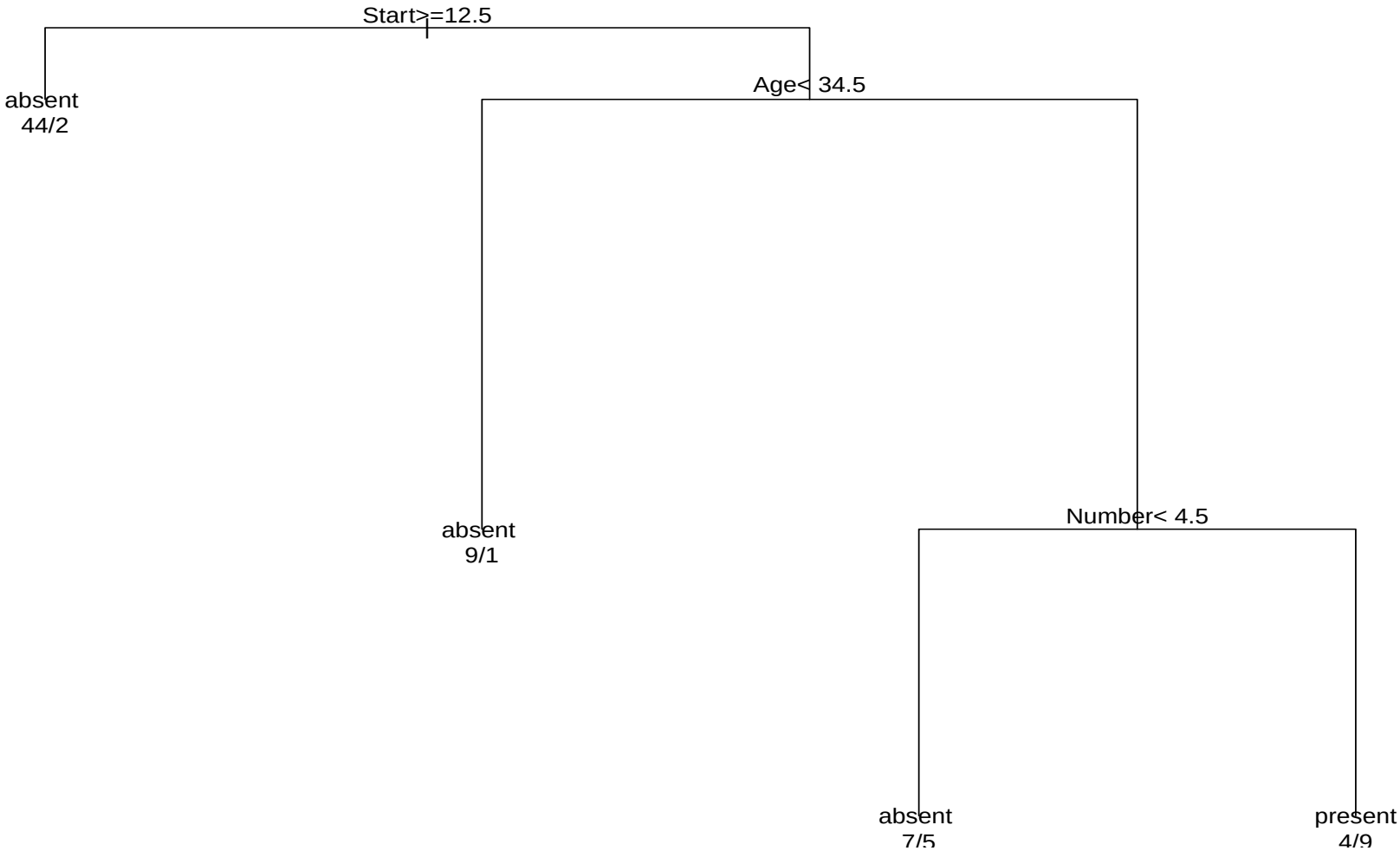
Classification Example - Kyphosis Case (count.)

```
tree_e <- rpart(Kyphosis ~ Age + Number + Start, data=kyphosis,
                parm=list(split="information"))
tree_e
```

```
## n= 81
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 81 17 absent (0.79012346 0.20987654)
##    2) Start>=12.5 46  2 absent (0.95652174 0.04347826) *
##    3) Start< 12.5 35 15 absent (0.57142857 0.42857143)
##      6) Age< 34.5 10  1 absent (0.90000000 0.10000000) *
##      7) Age>=34.5 25 11 present (0.44000000 0.56000000)
##       14) Number< 4.5 12  5 absent (0.58333333 0.41666667) *
##       15) Number>=4.5 13  4 present (0.30769231 0.69230769) *
```

Classification Example - Kyphosis Case (count.)

```
par(mar=c(1,1,1,1)); plot(tree_e); text(tree_e, use.n=TRUE)
```



Classification Example - Kyphosis Case

```
addmargins(table(kyphosis$Kyphosis, predict(tree_g, type="class")))
```

```
##
##             absent present Sum
##    absent       53      11  64
##    present       2      15  17
##    Sum          55      26  81
```

```
addmargins(table(kyphosis$Kyphosis, predict(tree_e, type="class")))
```

```
##
##             absent present Sum
##    absent       60       4  64
##    present       8       9  17
##    Sum          68      13  81
```

- Error-rates are similar but detail is not.(false positive, false negative)

Regression Example - CPU Case

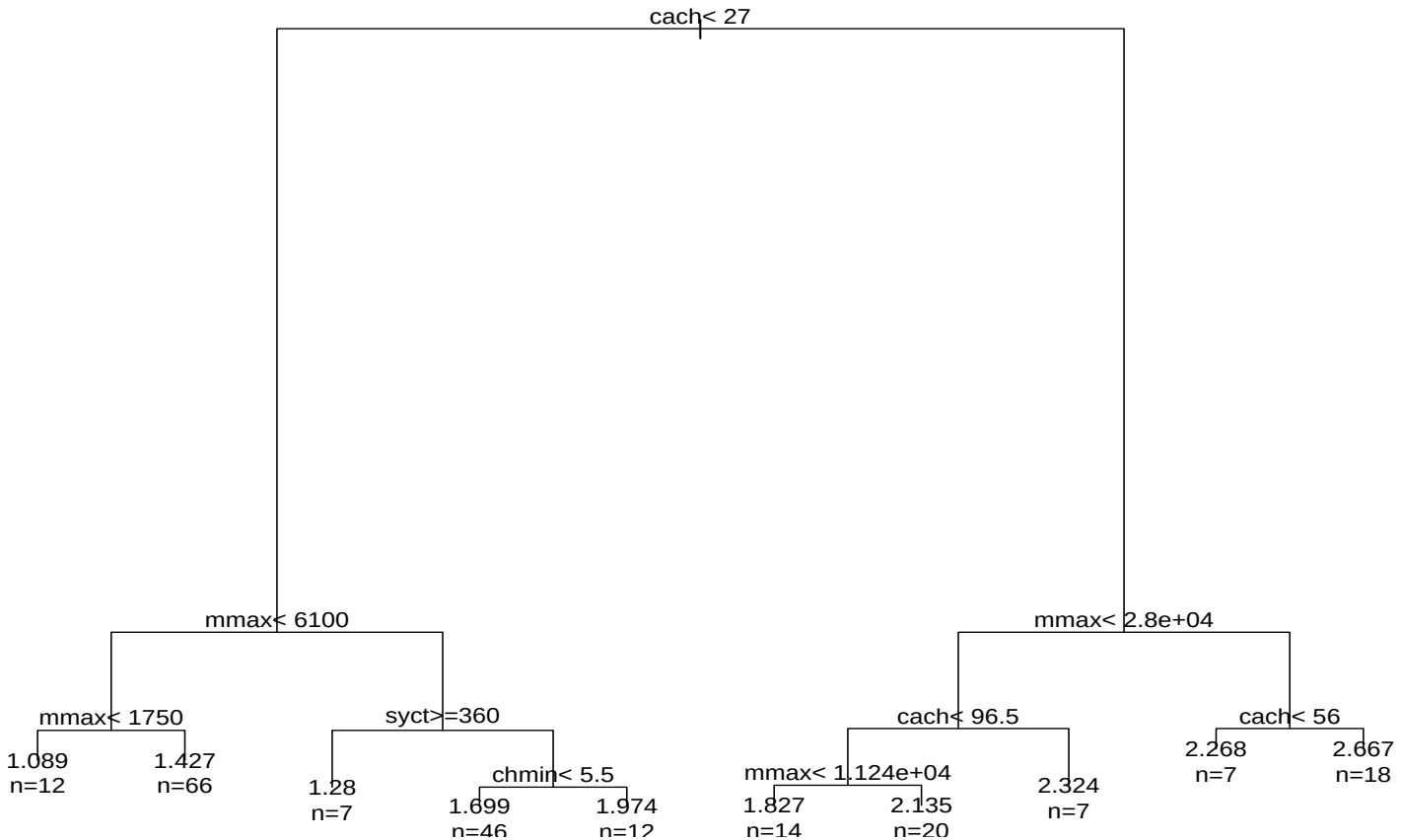
```
data(cpus, package="MASS")
str(cpus)
```

```
## 'data.frame':    209 obs. of  9 variables:
##  $ name   : Factor w/ 209 levels "ADVISOR 32/60",..: 1 3 2 4 5 6 8 9 10 7 ...
##  $ syct   : int  125 29 29 29 29 26 23 23 23 23 ...
##  $ mmin   : int  256 8000 8000 8000 8000 8000 16000 16000 16000 32000 ...
##  $ mmax   : int  6000 32000 32000 32000 16000 32000 32000 32000 64000 64000 ...
##  $ cach   : int  256 32 32 32 32 64 64 64 64 128 ...
##  $ chmin  : int  16 8 8 8 8 8 16 16 16 32 ...
##  $ chmax  : int  128 32 32 32 16 32 32 32 32 64 ...
##  $ perf   : int  198 269 220 172 132 318 367 489 636 1144 ...
##  $ estperf: int  199 253 253 253 132 290 381 381 749 1238 ...
```

- perf : Performance of CPU
- syct, mmin, mmax, cach, chmin, chmax : Features of CPU

Regression Example - CPU Case (count.)

```
tree_g_cpu <- rpart(log10(perf)~syct+mmin+mmax+cach+chmin+chmax,
                    data=cpus)
par(mar=c(1,1,1,1)); plot(tree_g_cpu); text(tree_g_cpu, use.n=TRUE)
```



Regression Example - CPU Case (count.)

```
pred_err <- log10(cpus$perf) - predict(tree_g_cpu)
mean(pred_err^2)
```

```
## [1] 0.03034244
```

- Mean Squared Error(MSE) : 0.03034244
- Should think about over-fitting.

Ensemble learning

- Ensemble learning use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the learning algorithms alone.
- Bagging, Randome Forest, Adaboost. . . ..

Bagging

- Bagging : Boostrap aggregation
  - 1. Do random sampling with replacement. (Boostrap)
  - 2. Then, built a model on that sample.
  - 3. Do many times 1 & 2 procedure.
  - 4. Final predictions are combined using voting(classification) or averaging(regression). (Aggregation)
    - Tree#1 : 49%, Tree#2 : 49%, Tree#3 : 99%, what is final predictions?
    - Hard voting : 1 vs 2 $\rightarrow$ Class 2
    - Soft voting : $\frac{1}{3}(49\% + 49\% + 99\%) = 60\% \rightarrow$ Class 1

Bagging Example - Kyphosis Case

```
library(ipred) # package for bagging
bag_kyphosis <- bagging(Kyphosis~Age+Number+Start,data=kyphosis)
bag_kyphosis
```

```
##
## Bagging classification trees with 25 bootstrap replications
##
## Call: bagging.data.frame(formula = Kyphosis ~ Age + Number + Start,
##     data = kyphosis)
```

```
addmargins(table(kyphosis$Kyphosis, predict(bag_kyphosis,
                                            type="class")))
```

```
##
##             absent present Sum
##    absent       55       9  64
##    present       9       8  17
##    Sum          64      17  81
```

Random Forest

- Random Forest is general version of bagging.
- Two key concepts that give it the name random :
  - A random sampling of training data set when building trees.(=bagging)
  - Random subsets of features considered when splitting each nodes. It makes various trees.

Random Forest Example - Kyphosis Case

```
library(randomForest) # package for RF
rf <- randomForest(Kyphosis~Age+Number+Start,data=kyphosis,
                   var.importance=TRUE)
rf
```

```
##
## Call:
##  randomForest(formula = Kyphosis ~ Age + Number + Start, data = kyphosis
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 19.75%
## Confusion matrix:
##         absent present class.error
## absent      61       3   0.0468750
## present     13       4   0.7647059
```

- Deault - Trees: 500, Variables tried at each split: 1 ($=\lfloor\sqrt{3}\rfloor$)

Random Forest Example - Kyphosis Case (count.)

```
rf$importance
```

```
##        MeanDecreaseGini
## Age            8.778802
## Number         5.318404
## Start          9.583841
```

- Variable's importance on Mean Decrease Gini : Start $>$ Age $>$ Number

Adaboost

- AdaBoost, short for "Adaptive Boosting", is the practical boosting algorithm proposed by Freund and Schapire in 1996.
- It focuses on classification problems and aims to convert a set of weak classifiers into a strong one.
    - Give bigger weight on misclassified observation, when do next random sampleing.

Adaboosting - Kyphosis Case

```
library(adabag) # package for adaboosting
adaboost <- boosting(Kyphosis~Age+Number+Start,
                     data=kyphosis, mfinal=20)
addmargins(table(kyphosis$Kyphosis,
                 predict(adaboost, newdata=kyphosis)$class))
```

```
##
##              absent present Sum
##    absent        64       0  64
##    present        0      17  17
##    Sum           64      17  81
```

- Error is zero.

```
adaboost$importance
```

```
##       Age    Number     Start
## 42.39695 23.06030 34.54275
```

- Variable's importance : Age $>$ Start $>$ Number (vs RF)

Real Consulting Case - Spring, 2019

- Client : Researcher of Ministry of Health and Welfare.
- Subject : Finding important variables for 'Undiagnosis of alcohol abuse' based on Machine Learning.
- Data size : 23,197(N) x 20(p)
  - N : Person who answered "Drink alcohol, almost everyday"
  - p : Features of body, living
- Used Package : randomForest of R

Real Consulting Case - Spring, 2019 (cont.)

- Result : Top 7 important variables

Variable	MeanDecreaseGini
Test year	1775.64395
Disability type	714.54617
Disability severity	659.39391
Drinking quantity	379.69569
CCI categorization	335.88330
Age group(Numeric)	316.30588
Residential district	310.21100

Reference

- Heo(2014), Applied Data Analysis Using R, Ch.18
- Group 4(Spring, 2019), Final Report

Q & A

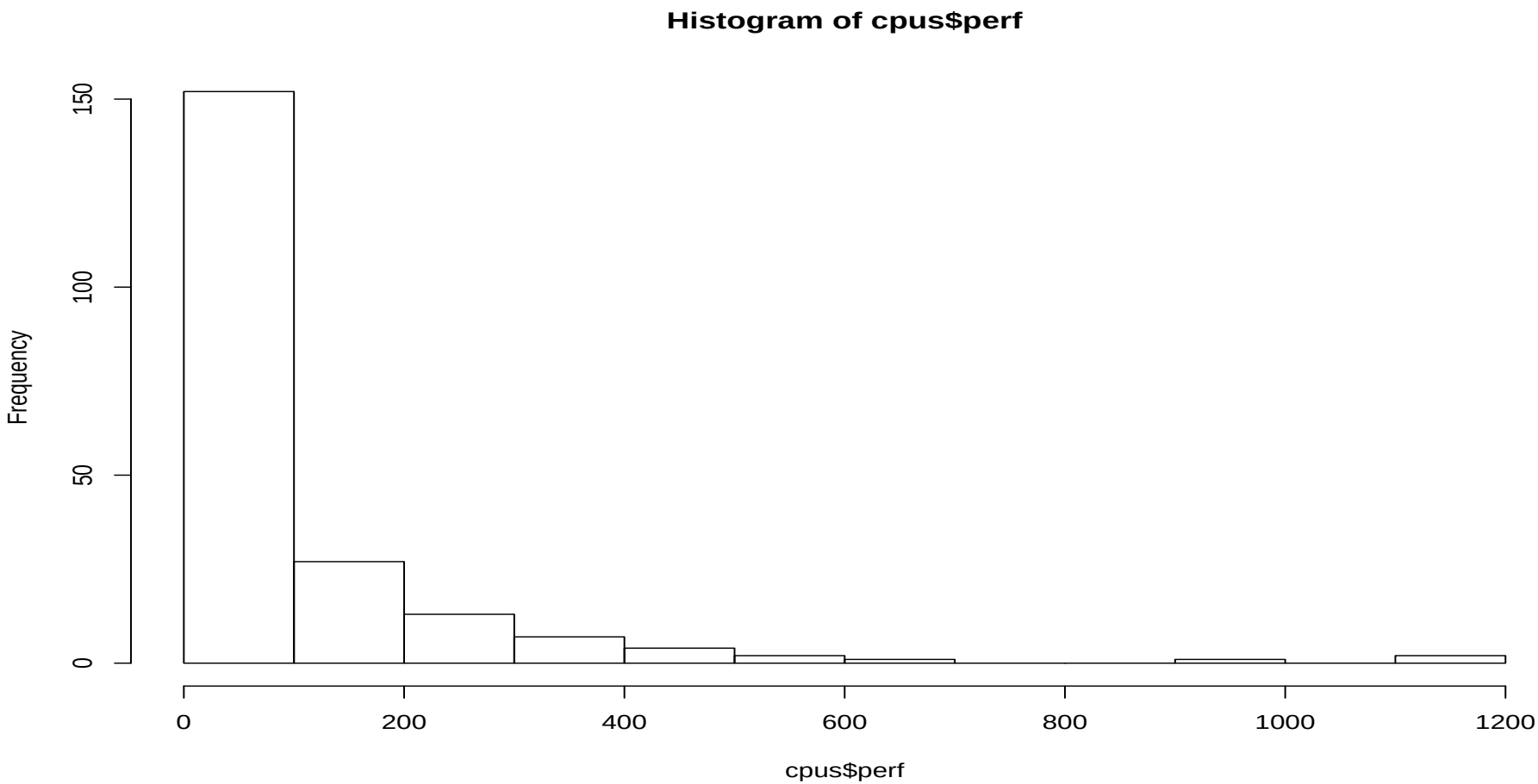
Appendix - Weighted decision tree

```
kyphosis$wts <- ifelse(kyphosis$Kyphosis=="present", 0.79, 0.21)
tree_e_w <- rpart(Kyphosis ~ Age+Number+Start, data=kyphosis,
                  parms=list(split="information"),
                  control=rpart.control(cp=0.05), weights=wts)
table(kyphosis$Kyphosis, predict(tree_e_w, type="class"))
```

```
##
##              absent present
##    absent        51      13
##    present        2      15
```

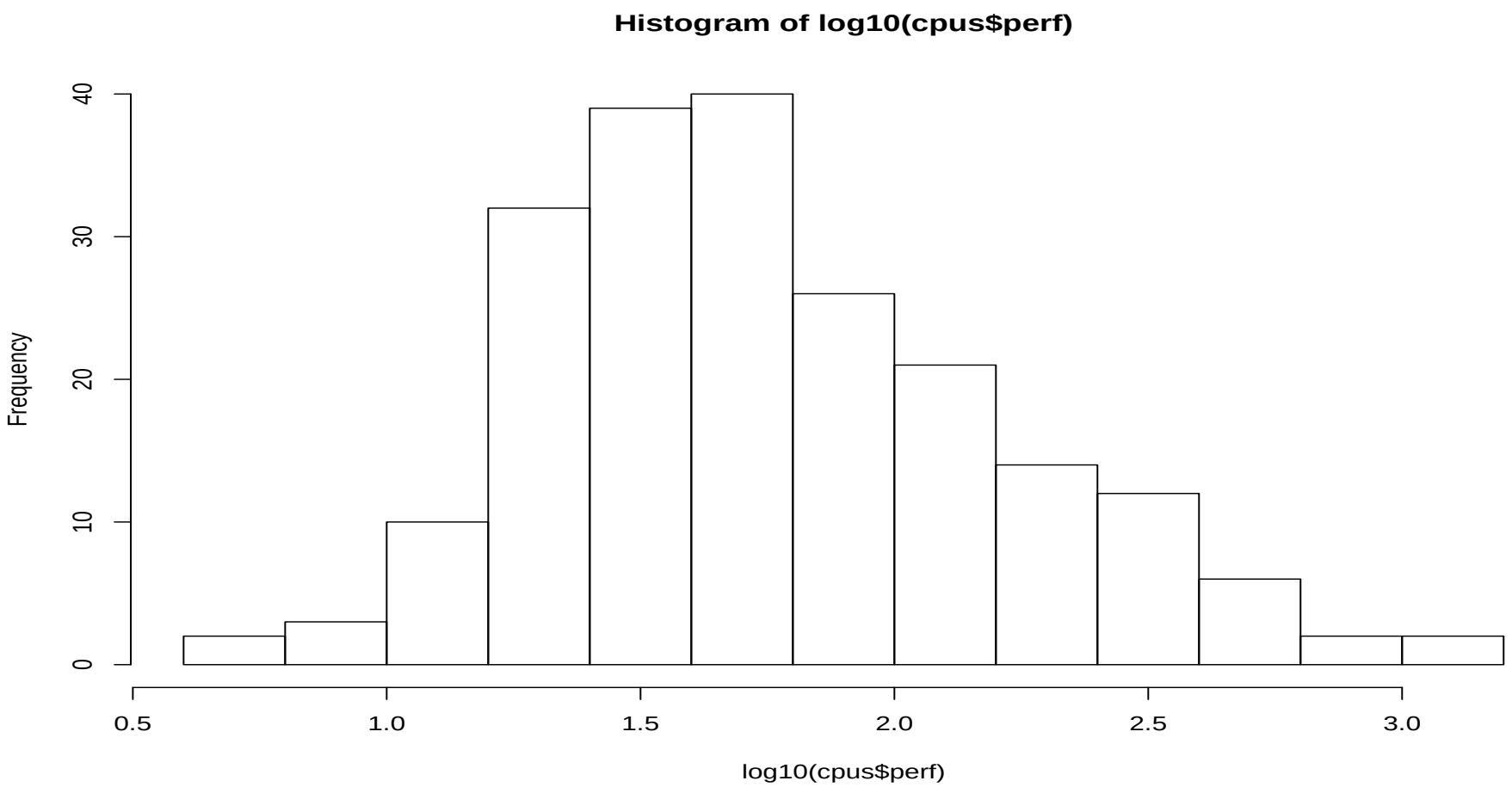
Appendix - log10(perf)

```
hist(cpus$perf)
```



Appendix - log10(perf)

```
hist(log10(cpus$perf))
```



Appendix - Dealing Imbalance

```
kyphosis.present <- kyphosis[kyphosis$Kyphosis=="present",]
kyphosis.absent <- kyphosis[kyphosis$Kyphosis=="absent",]
kyphosis.balaced <- rbind(kyphosis.present, kyphosis.present,
                          kyphosis.present, kyphosis.present,
                          kyphosis.absent)
table(kyphosis.balaced$Kyphosis)
```

```
##
##  absent present
##      64      68
```

Appendix - Split train and test dataset

```
subsample <- sample(1:209, replace=TRUE)
cpus_train <- cpus[subsample,]
cpus_test <- cpus[-subsample,]
cpus_tree_2 <- rpart(log10(perf)~syct+mmin+mmax+cach+chmin+chmax,
                     data=cpus_train)
pred_err_2 <- log10(cpus_test$perf)-predict(cpus_tree_2,
                                            newdata=cpus_test)
mean(pred_err_2^2)
```

```
## [1] 0.06635754
```