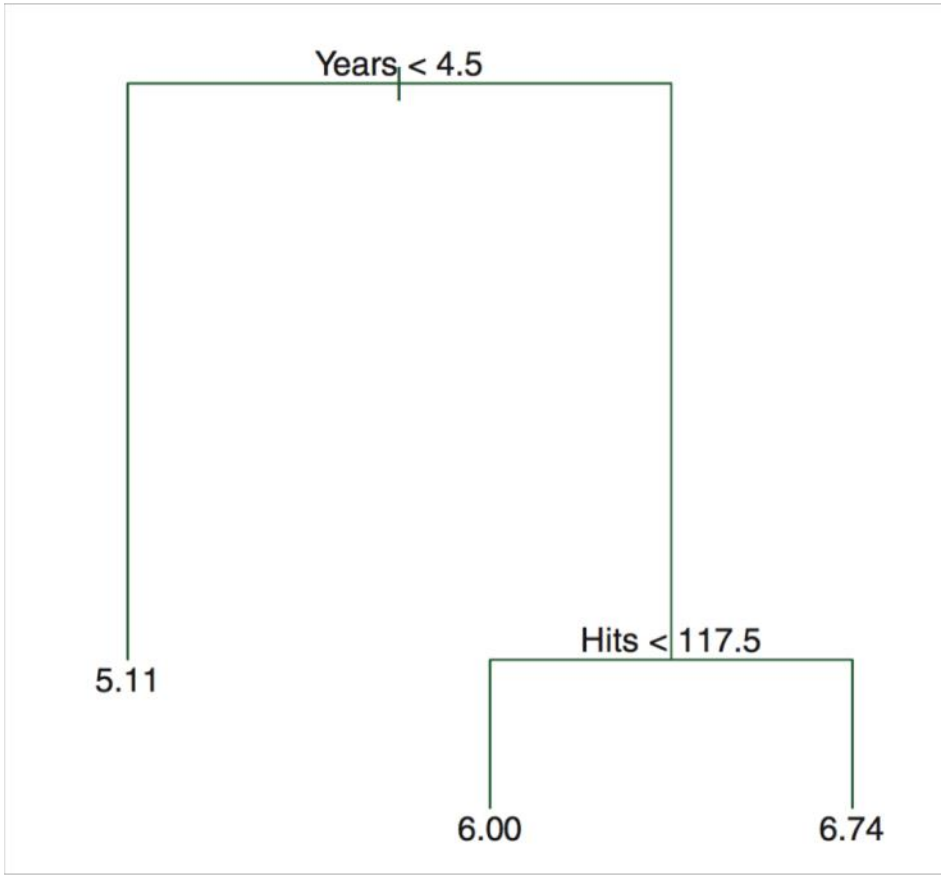
Tree Based Methods (Chapter 8 in ISLR)

Belize Machine Learning Training

Jonathan Hersh (Chapman Univesity)

Regression Trees

- **Tree based methods *stratify* or *segment* the predictor space into different regions**
- Regions are stratified via simple rules
- The splitting rules can be summarized into a tree that is very intuitive



Pros and Cons of Trees

Pros

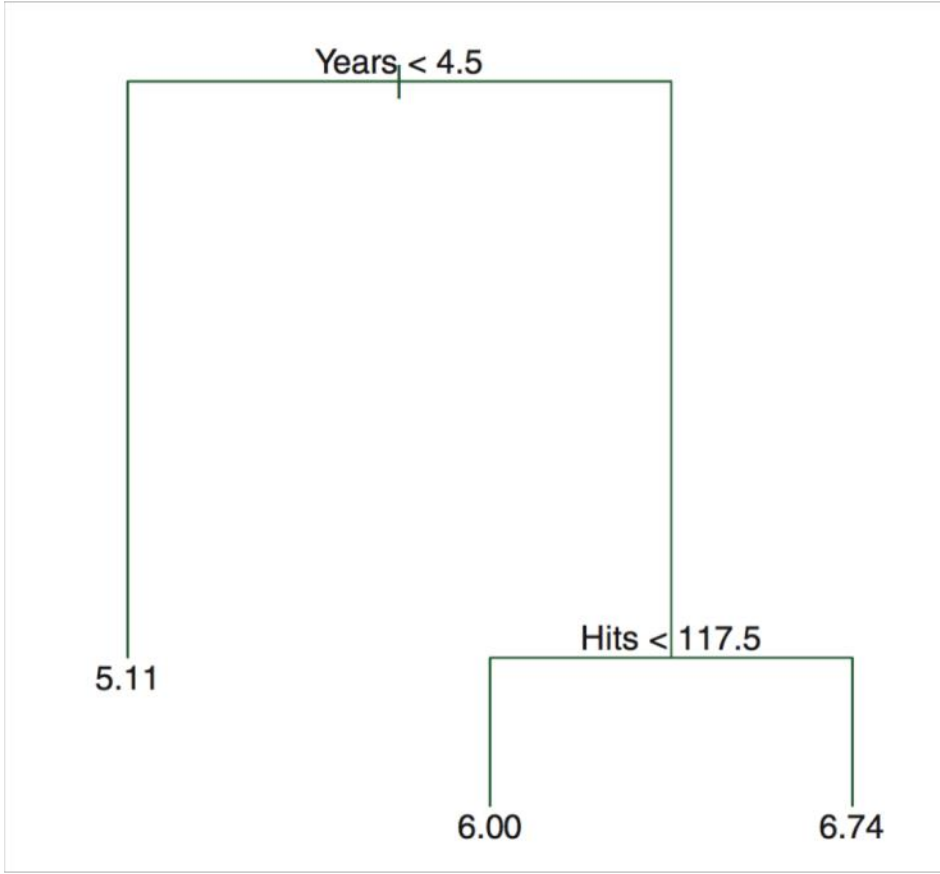
- Simple
- Easy to interpret
- Easy to explain
- Can be displayed graphically!
- Bagging, boosting, and random forests very powerful (combining trees)

Cons

- Slow with large datasets
- Not easy to use “out of the box”
- Choice of split can be unstable

Decision/Regression Trees

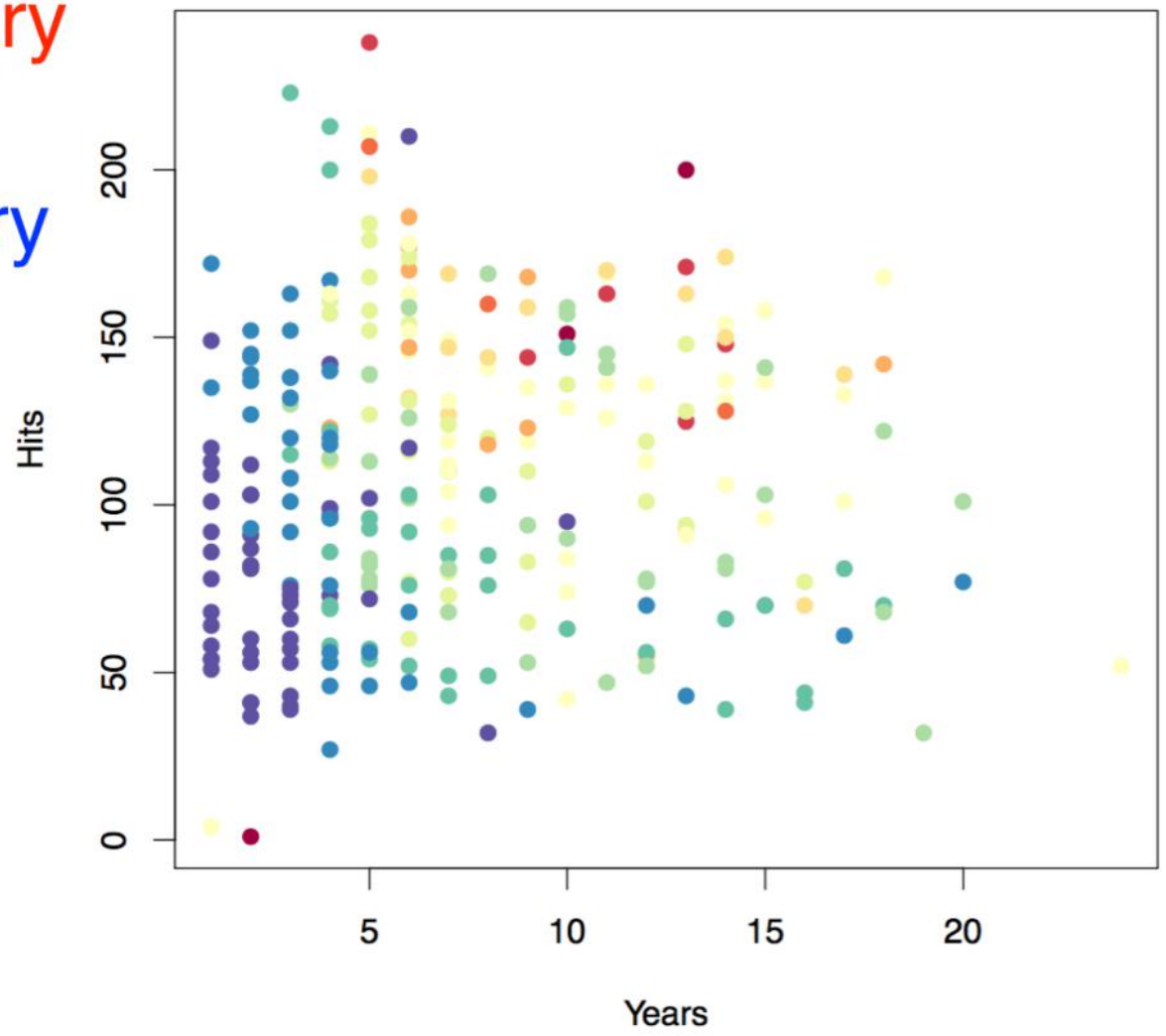
- Decision trees can be applied to both regression problems ($y_i \in R$) and classification problems $y_i \in \{class1, class2, \dots, \}$
- We'll consider both



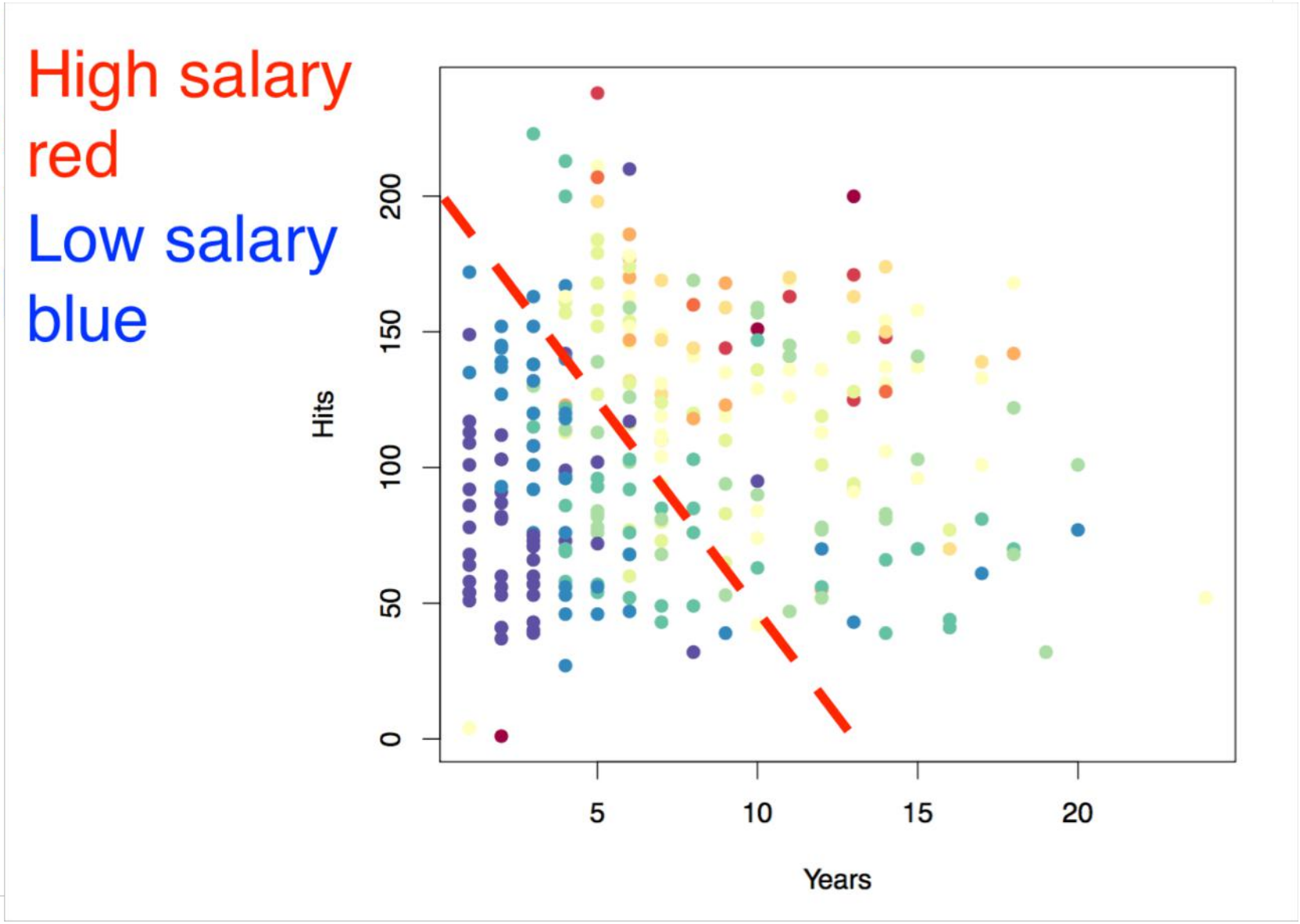
Baseball salary data: how to partition/stratify?

High salary
red

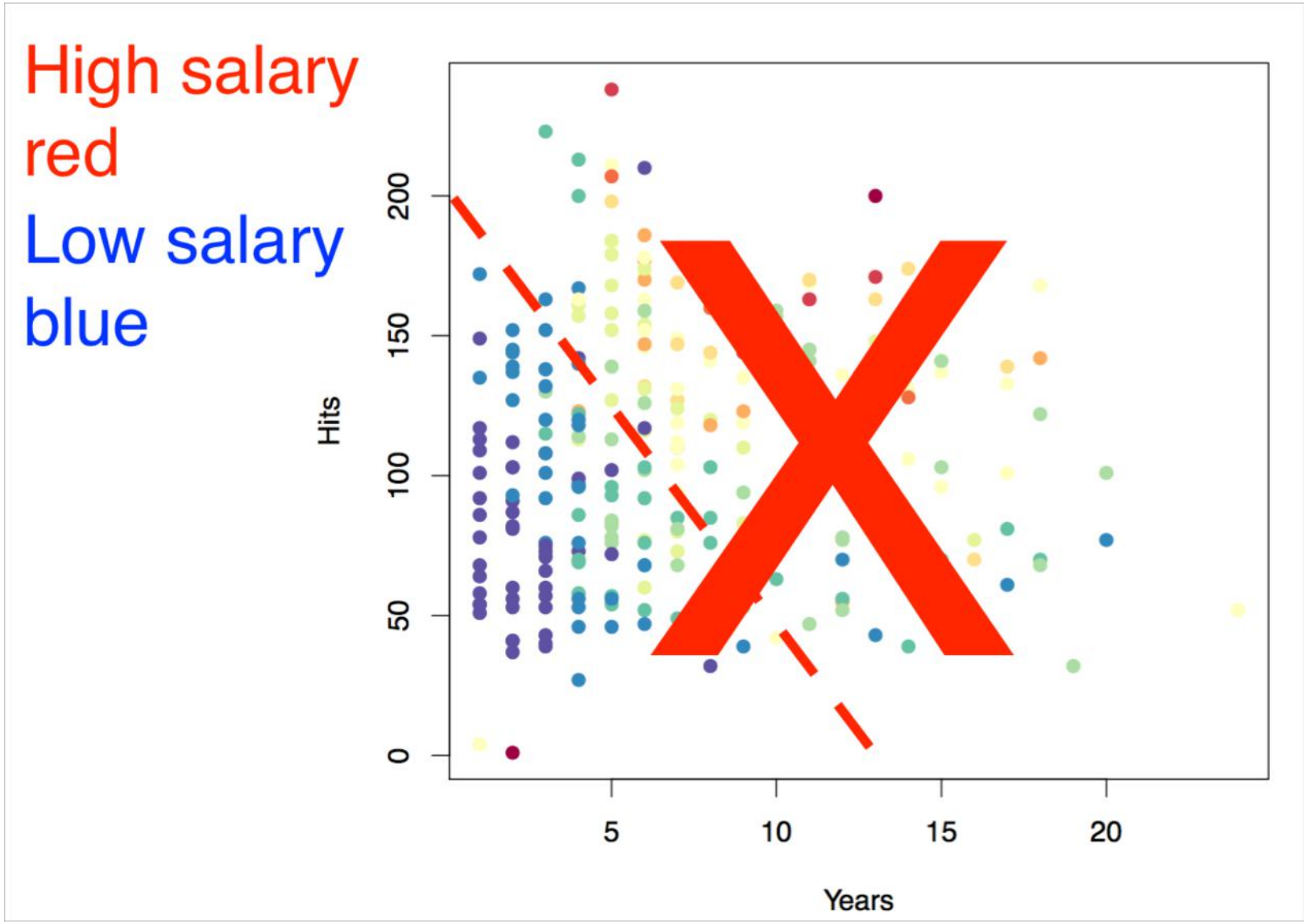
Low salary
blue



Baseball salary data: how to partition/stratify?

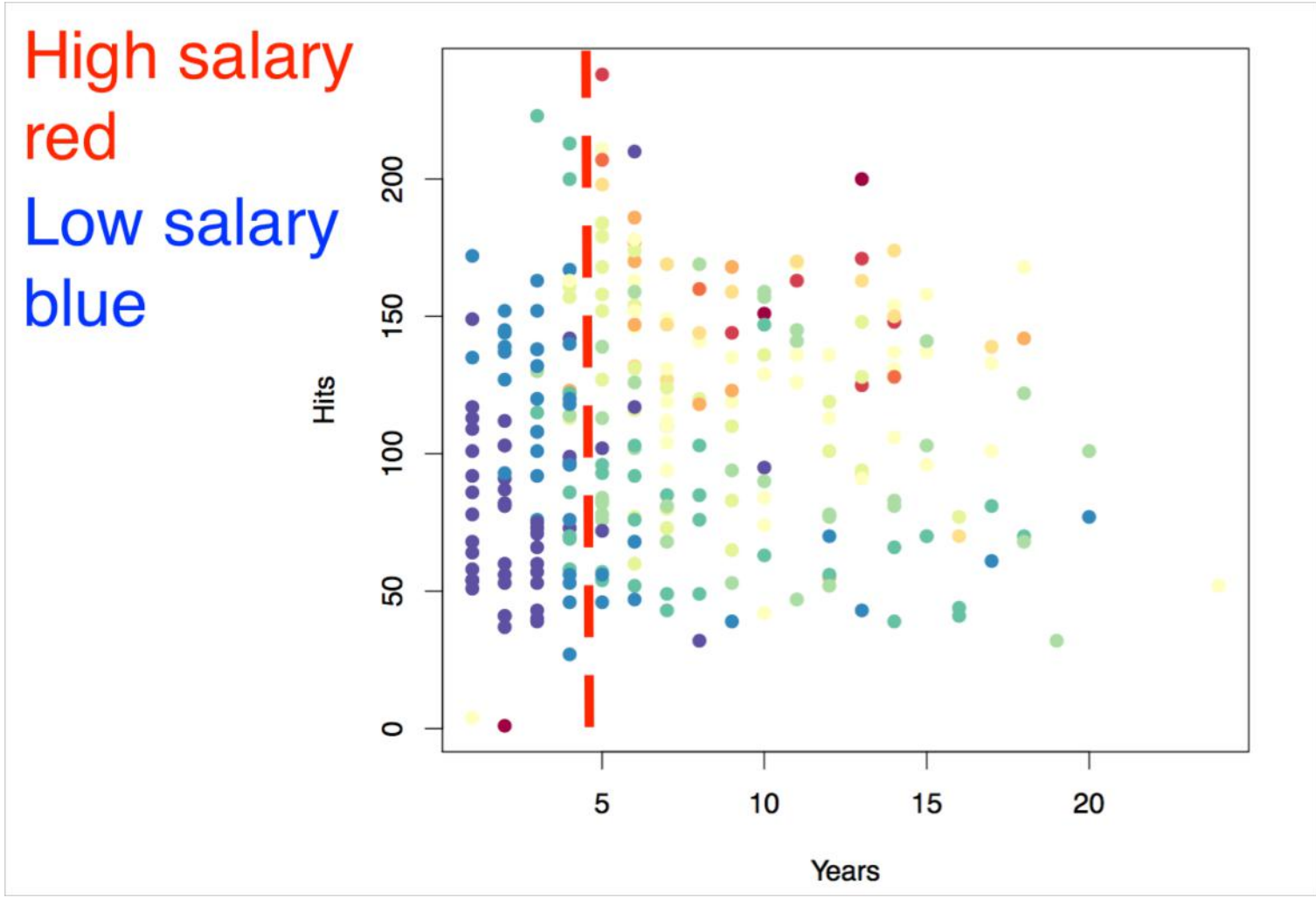


Baseball salary data: how to partition/stratify?



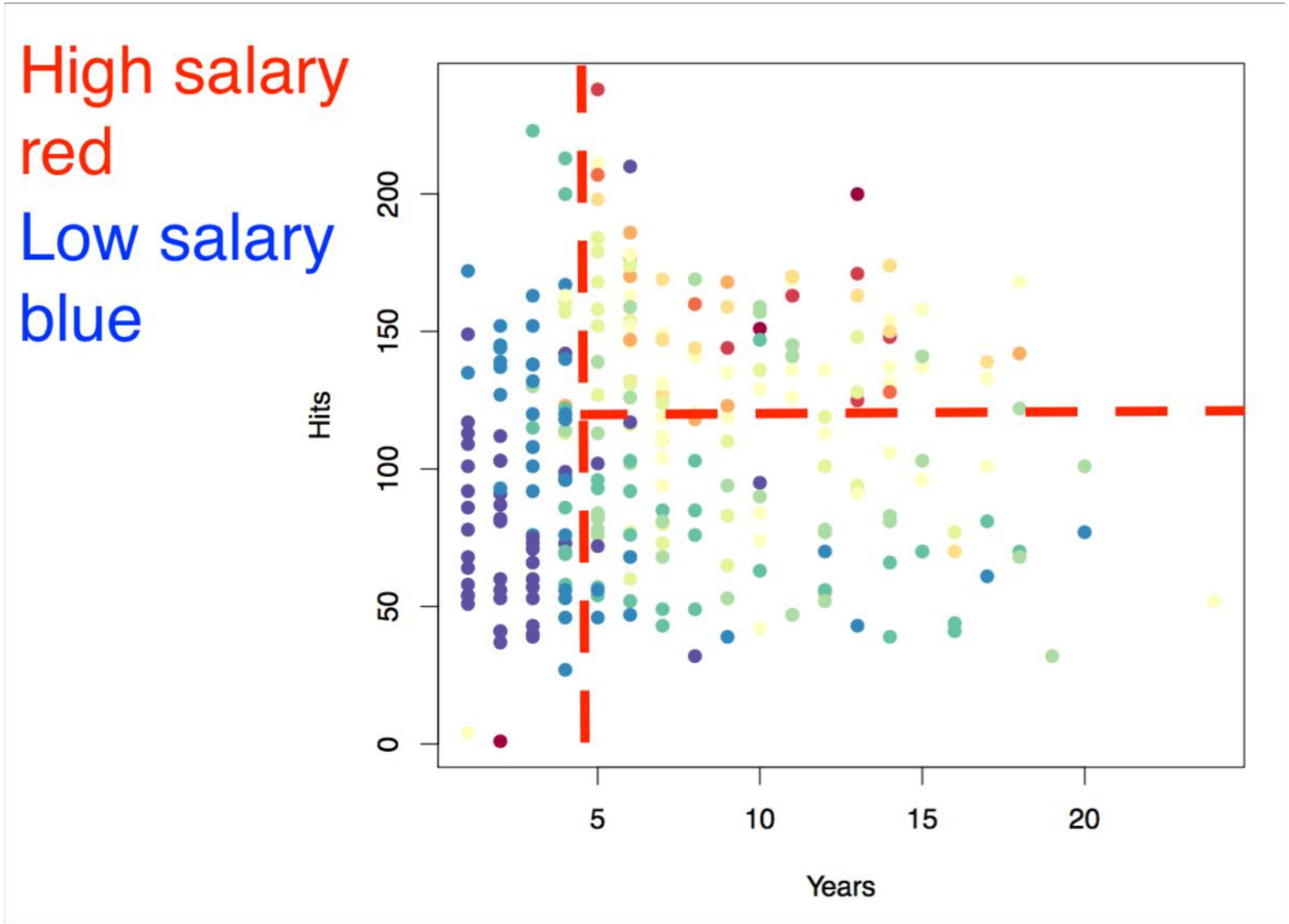
- Only linear classification rules are allowed, e.g. year > 10

Baseball salary data: split 1



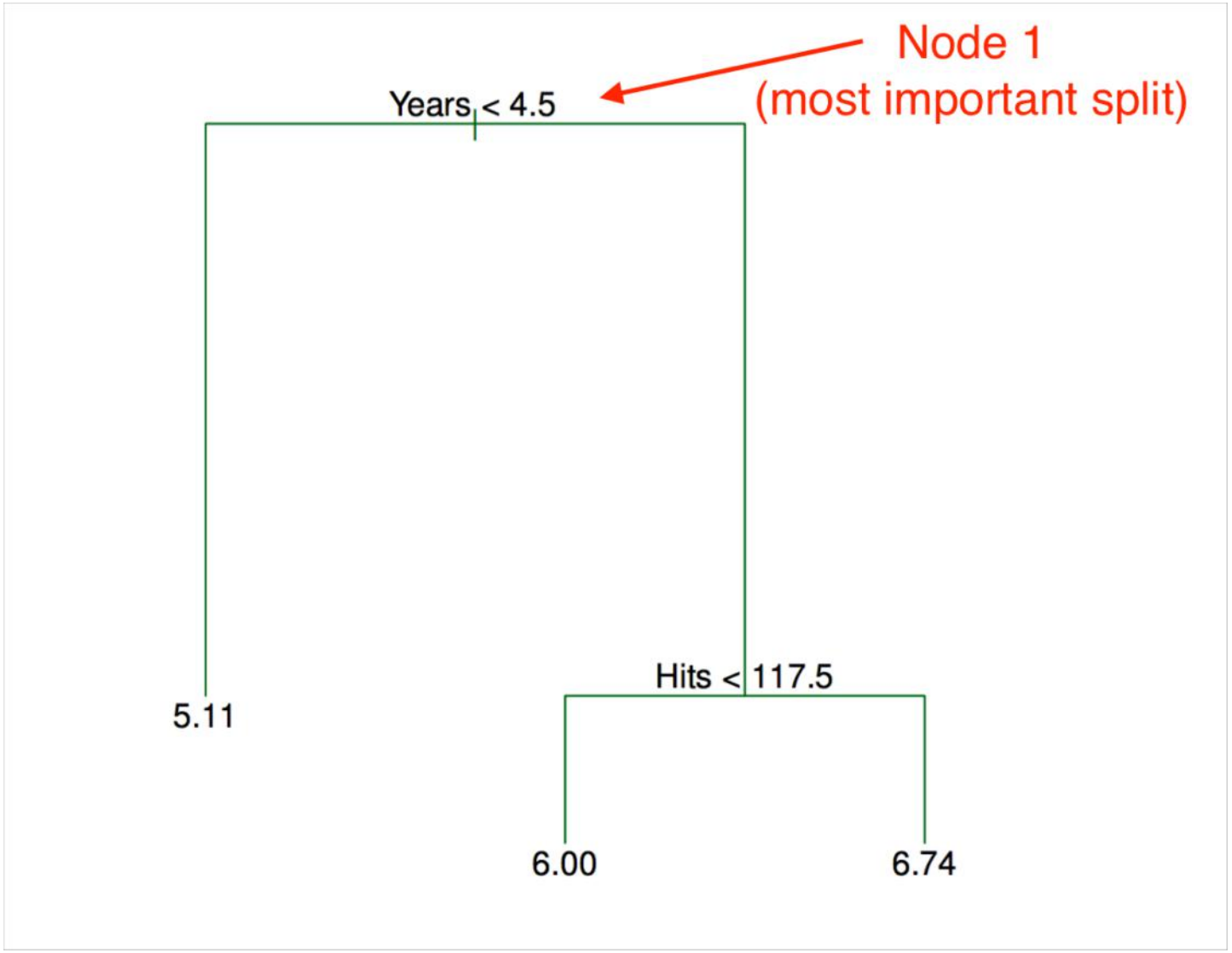
- Split 1: years > 4.5

Baseball salary data: split 2



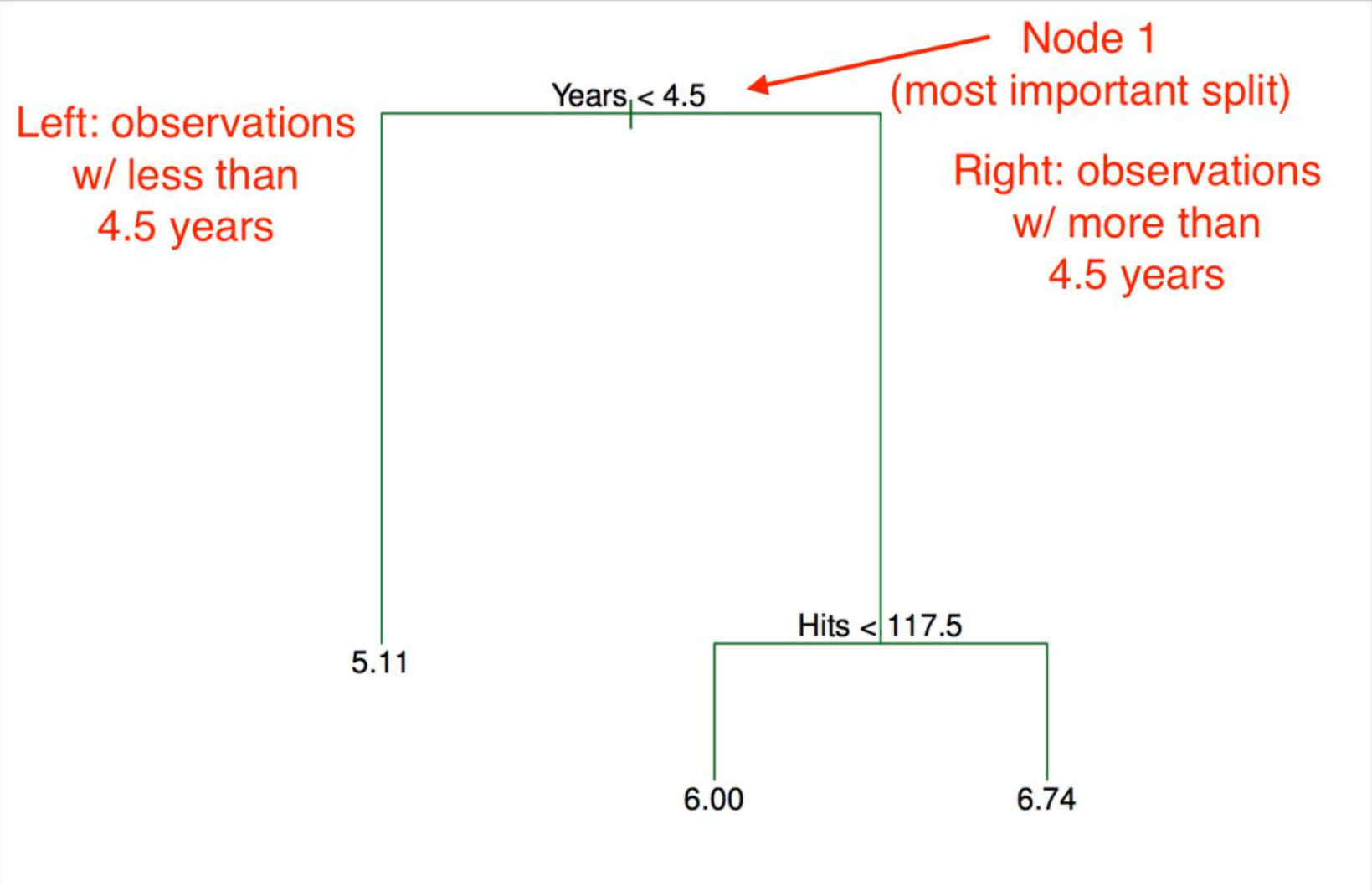
- Split 1: years > 4.5
- Split 2: hits > 117.5

Tree Representation

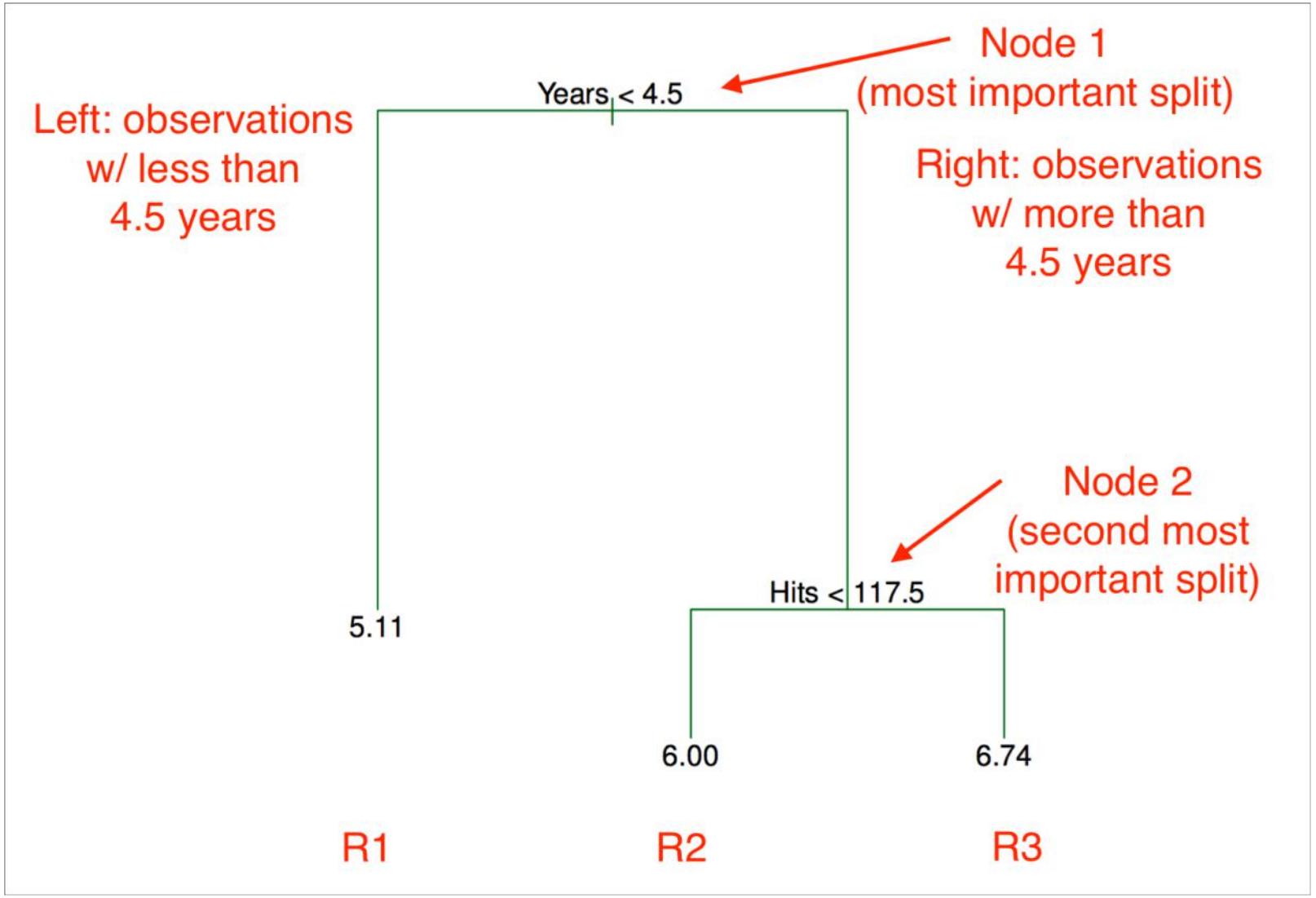


- Trees are read top-down
- Most important split is at top
- Length represents how much within-cluster variance decreases from split

Tree Representation



Tree Representation



- At the end of the tree are "leafs"
- For regression, avg value of observations in leaf => $\hat{y}$ for observation

Details of tree building algorithm

- Computationally infeasible to consider all combinations of splits
- **Instead, use a top-down, greedy approach called recursive binary splitting**
- Greedy here means at each step, we only consider the best split, without caring how it affects successive nodes
- “top down” because we start with the best split and proceed downward (no backing up)

Pruning trees

- How do we know when to stop splitting the data?
- **Trees with many splits can overfit the data**
- Solution is to grow a large tree T_0, then prune it to obtain a smaller sub-tree

Decision tree algorithm (8.1 in ISLR)

1. Use recursive binary splitting to grow a large tree on the training data
2. Apply pruning to large tree to obtain sequence of subtrees, α_1, α_2
3. Use *k*-fold cross validation to find $\hat{\alpha}$. For each *k* = 1, . . . ,*K* :
    1. Repeat steps 1 and 2 on all but the *k*-th fold of training data
    2. Evaluate mean squared prediction error on data in left-out *k*-th fold
    3. Average the results for each value of α. Choose the value of α that minimizes error α
4. Return the subtree that minimizes cross-validated error

Baseball example, unpruned tree

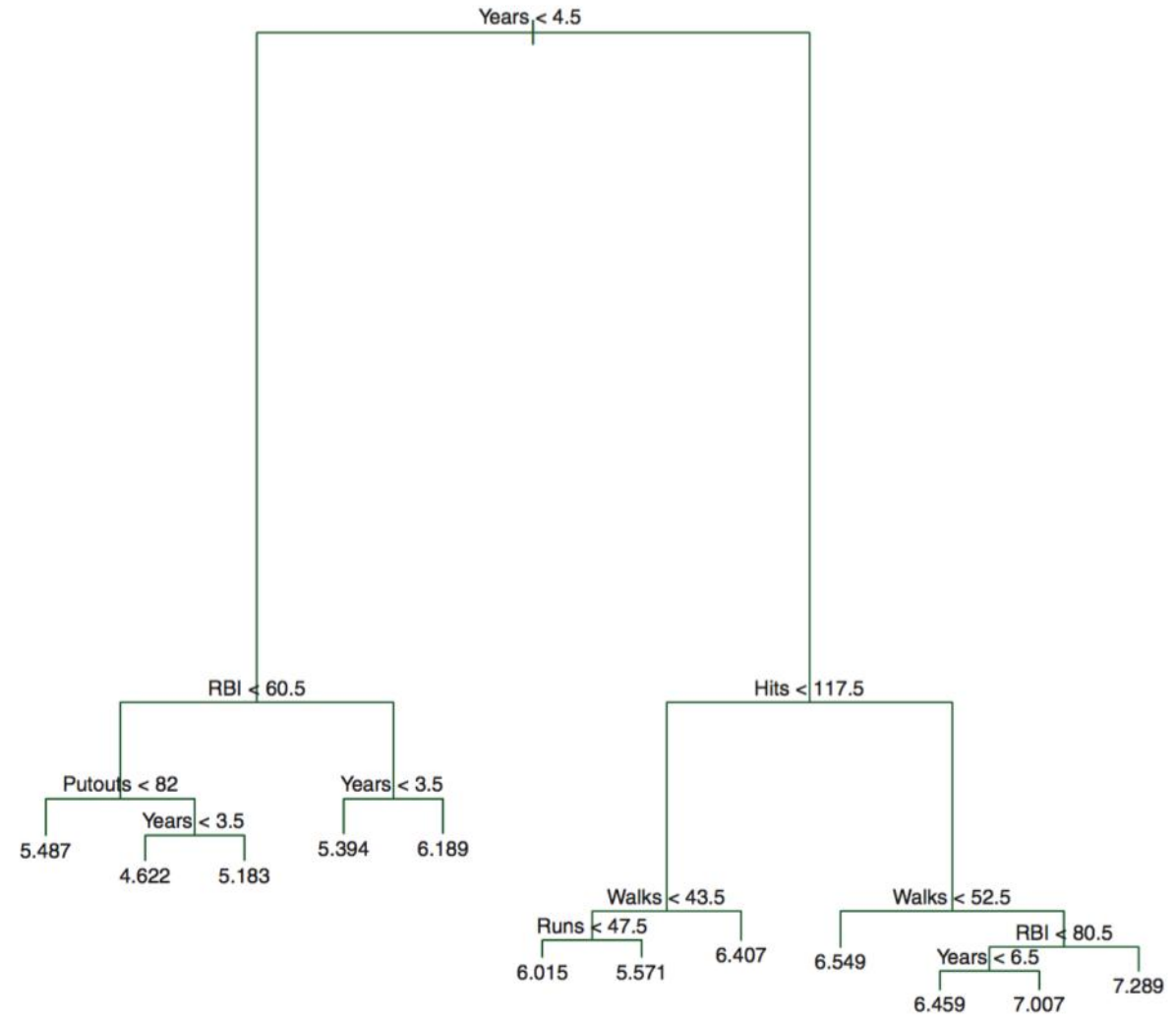


FIGURE 8.4. *Regression tree analysis for the* `Hitters` *data. The unpruned tree that results from top-down greedy splitting on the training data is shown.*

Cross-validate to find alpha

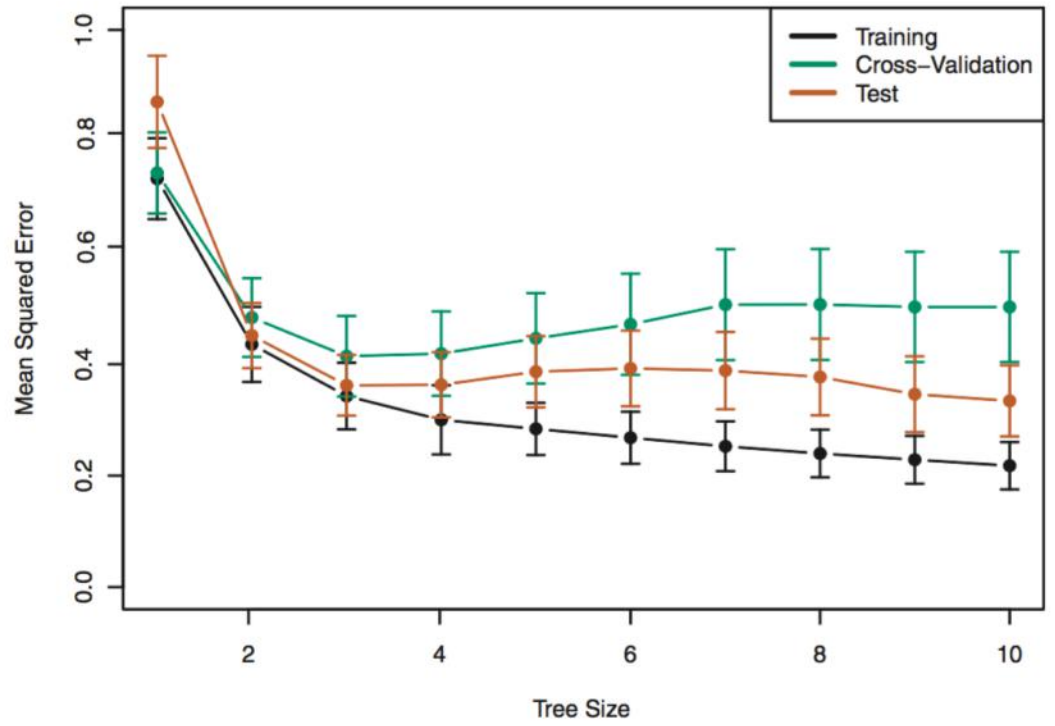
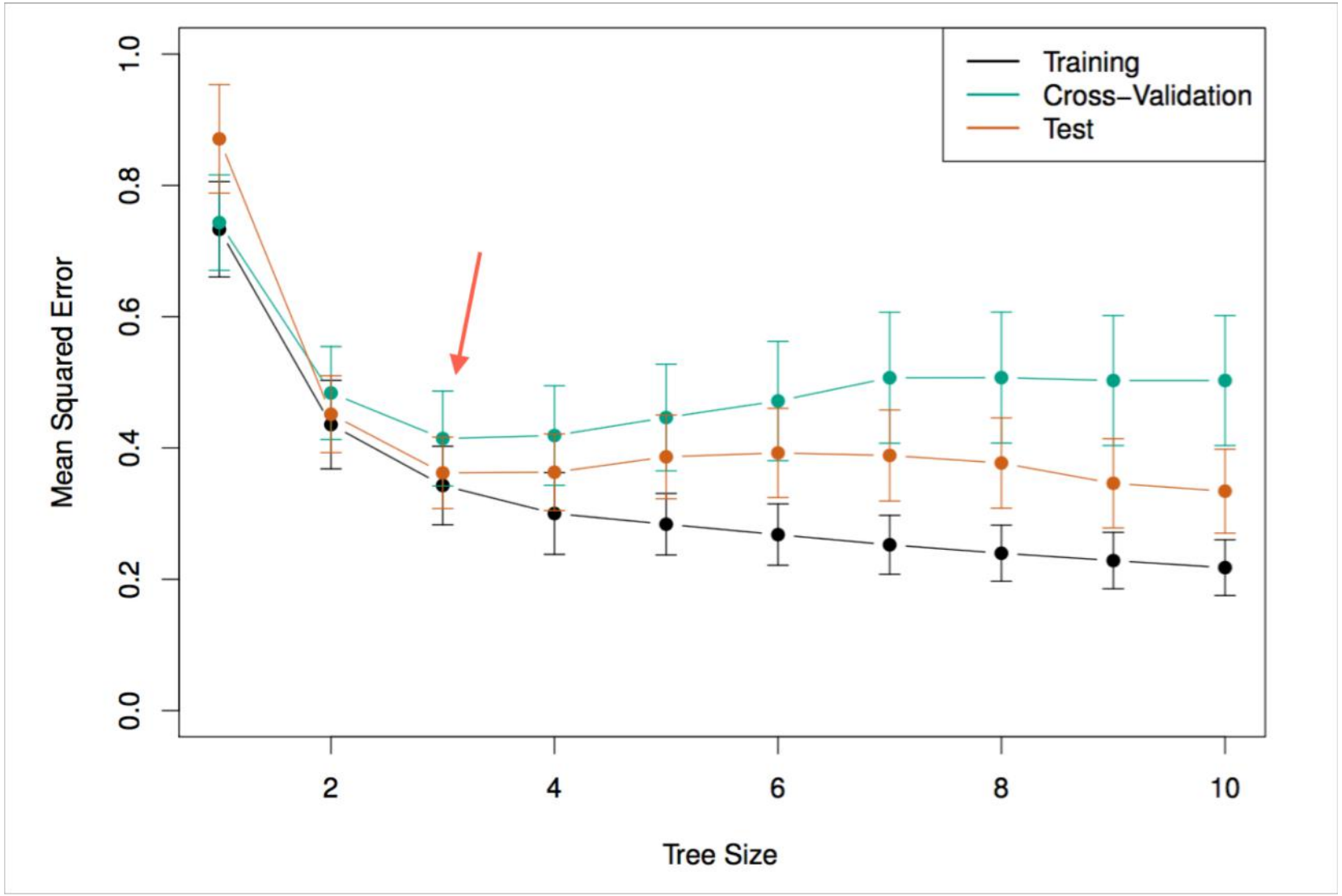


FIGURE 8.5. *Regression tree analysis for the* `Hitters` *data. The training, cross-validation, and test MSE are shown as a function of the number of terminal nodes in the pruned tree. Standard error bands are displayed. The minimum cross-validation error occurs at a tree size of three.*

Cross-validate to find alpha



Carseats dataset in ISLR

Carseats {ISLR}

Sales of Child Car Seats

Description

A simulated data set containing sales of child car seats at 400 different stores.

Usage

```
Carseats
```

Format

A data frame with 400 observations on the following 11 variables.

`Sales`

Unit sales (in thousands) at each location

`CompPrice`

Price charged by competitor at each location

`Income`

Community income level (in thousands of dollars)

`Advertising`

Local advertising budget for company at each location (in thousands of dollars)

`Population`

Population size in region (in thousands)

`Price`

tree() function in package “tree” to build regression tree

tree {tree} R Documentation

Fit a Classification or Regression Tree

Description

A tree is grown by binary recursive partitioning using the response in the specified formula and choosing splits from the terms of the right-hand-side.

Usage

```
tree(formula, data, weights, subset,
     na.action = na.pass, control = tree.control(nobs, ...),
     method = "recursive.partition",
     split = c("deviance", "gini"),
     model = FALSE, x = FALSE, y = TRUE, wts = TRUE, ...)
```

Arguments

`formula`	A formula expression. The left-hand-side (response) should be either a numerical vector when a regression tree will be fitted or a factor, when a classification tree is produced. The right-hand-side should be a series of numeric or factor variables separated by `+`; there should be no interaction terms. Both `.` and `-` are allowed: regression trees can have `offset` terms.
`data`	A data frame in which to preferentially interpret `formula`, `weights` and `subset`.
`weights`	Vector of non-negative observational weights; fractional weights are allowed.
`subset`	An expression specifying the subset of cases to be used.
`na.action`	A function to filter missing data from the model frame. The default is `na.pass` (to do nothing) as `tree` handles missing values (by dropping them down the tree as far as possible).
`control`	A list as returned by `tree.control`.
`method`	character string giving the method to use. The only other useful value is `"model.frame"`.
`split`	Splitting criterion to use.
`model`	If this argument is itself a model frame, then the `formula` and `data` arguments are ignored, and `model` is used to define the

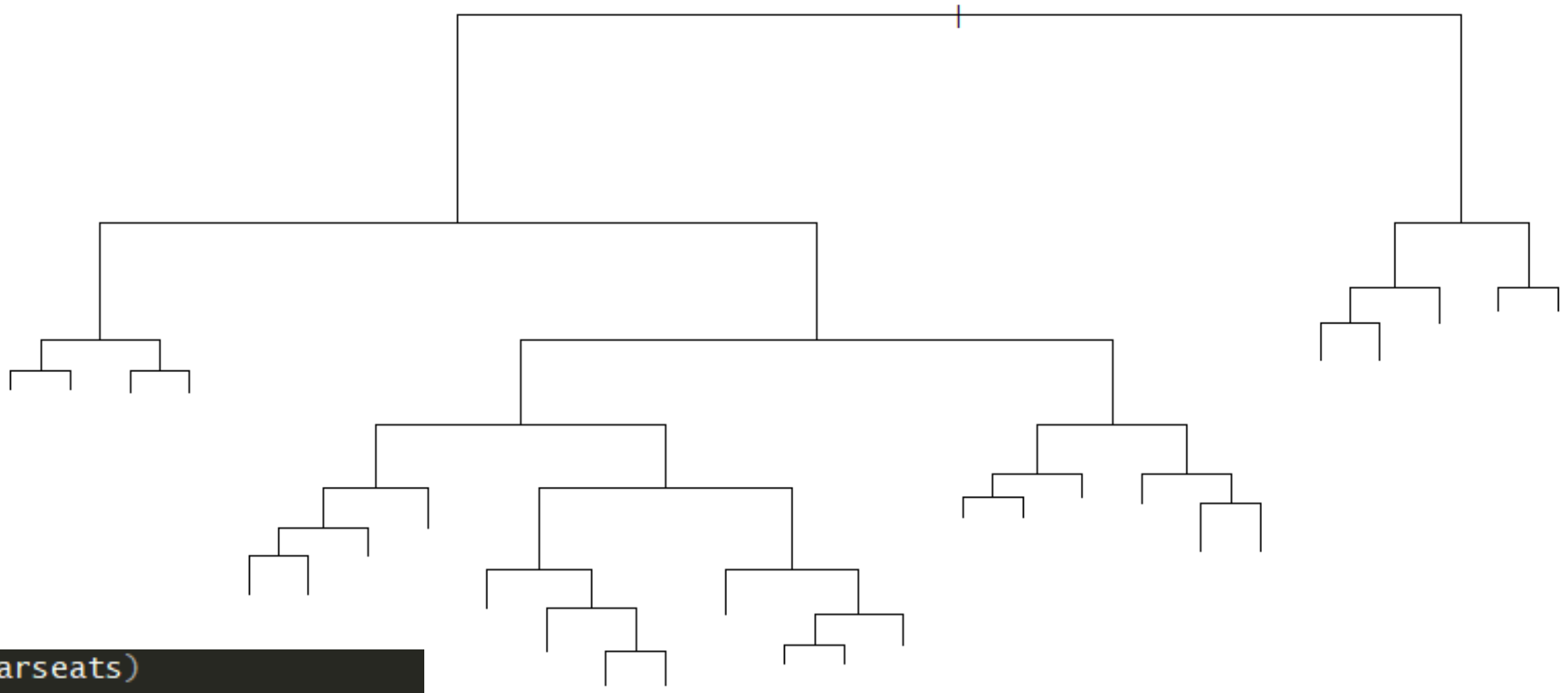
Estimate a tree model to predict carseat sales

```
require('ISLR')
require('tree')
attach(Carseats)
hist(Sales)

# create a high/low sales indicator
High = ifelse(Sales<=8,"No","Yes")
Carseats=data.frame(Carseats,High)
# estimate a tree model to predict whether
# carseat sales are high
tree.carseats = tree(High~.-Sales,data=Carseats)
# summarize tree
summary(tree.carseats)
```

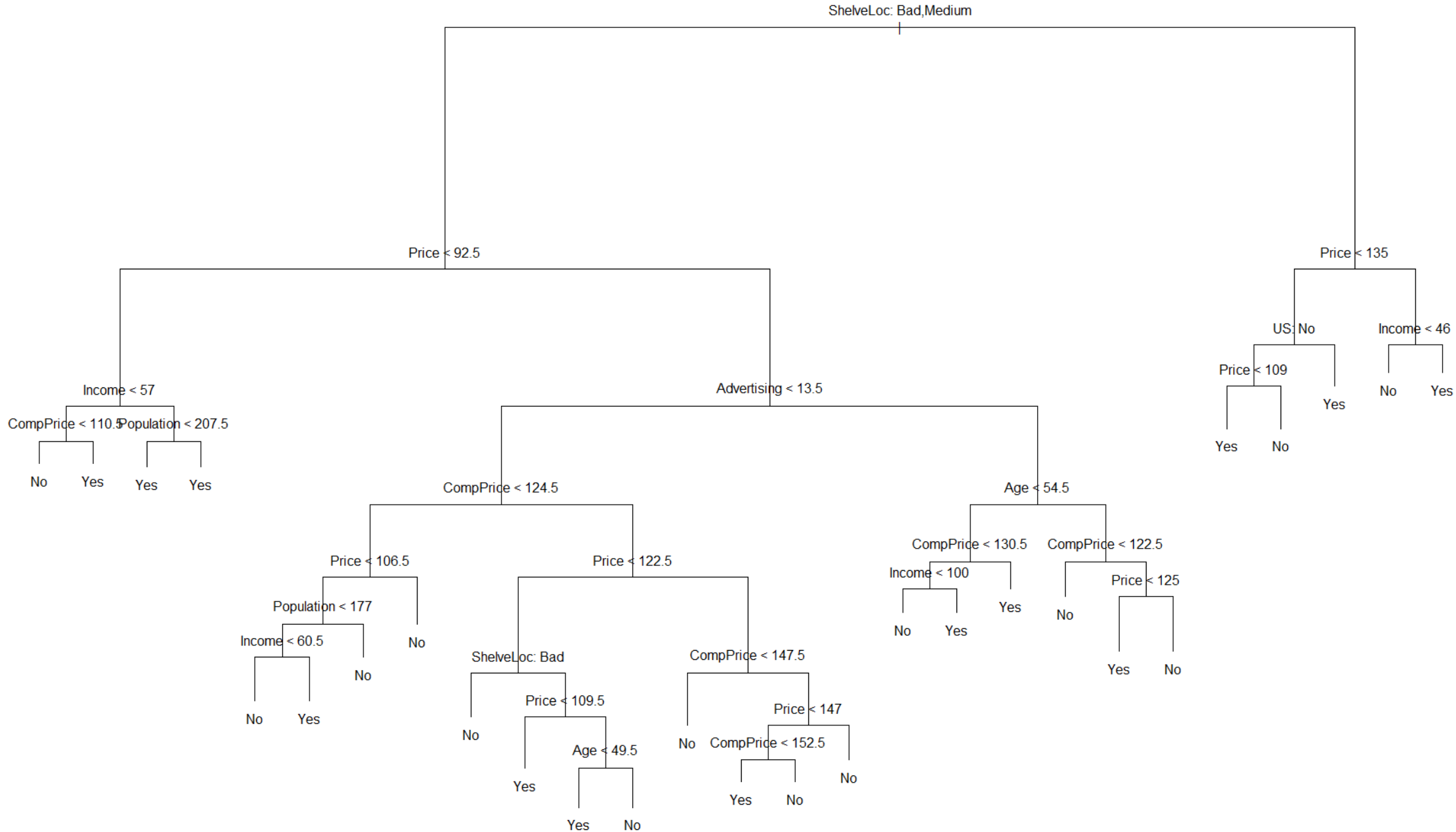
Estimate a tree model to predict carseat sales

```
> summary(tree.carseats)

Classification tree:
tree(formula = High ~ . - Sales, data = Carseats)
Variables actually used in tree construction:
[1] "ShelveLoc"   "Price"       "Income"      "CompPrice"   "Population"
"Advertising" "Age"
[8] "US"
Number of terminal nodes:  27
Residual mean deviance:  0.4575 = 170.7 / 373
Misclassification error rate: 0.09 = 36 / 400
```

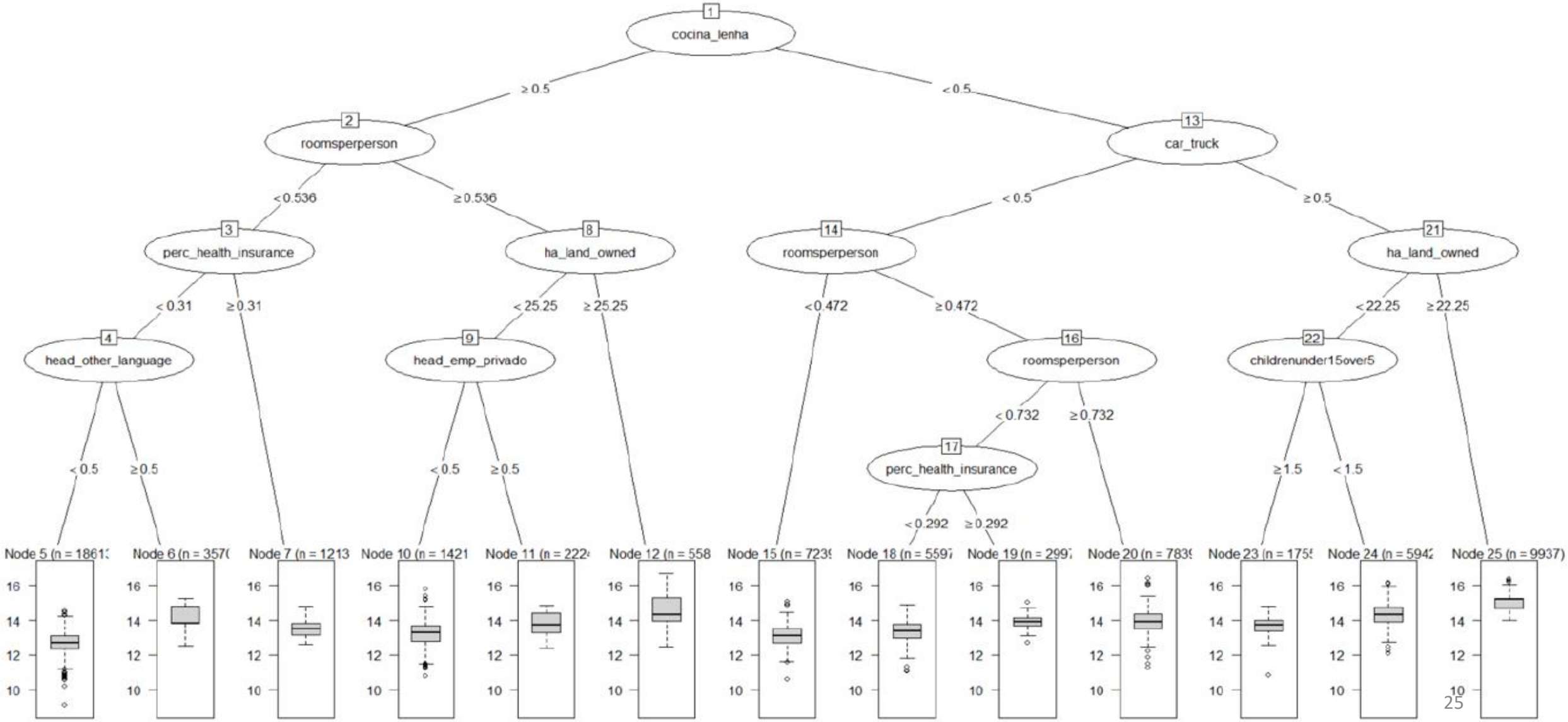
Plot tree



```
37  plot(tree.carseats)
38
39  text(tree.carseats,pretty=0)
40
```


ShelveLoc: Bad,Medium
Price < 92.5
Price < 135
Income < 57
Advertising < 13.5
US: No
Income < 46
CompPrice < 110.5
Population < 207.5
CompPrice < 124.5
Age < 54.5
Price < 109
No
Yes
Yes
Yes
Price < 106.5
Price < 122.5
CompPrice < 130.5
CompPrice < 122.5
Yes
No
No
Yes
Population < 177
Income < 100
Price < 125
Income < 60.5
No
ShelveLoc: Bad
CompPrice < 147.5
No
Yes
No
Yes
Yes
No
No
Yes
Price < 109.5
Price < 147
No
Age < 49.5
No
CompPrice < 152.5
Yes
No
Yes
No
No

Example tree to predict poverty in Paraguay



Netflix prize

Leaderboard

10.05%

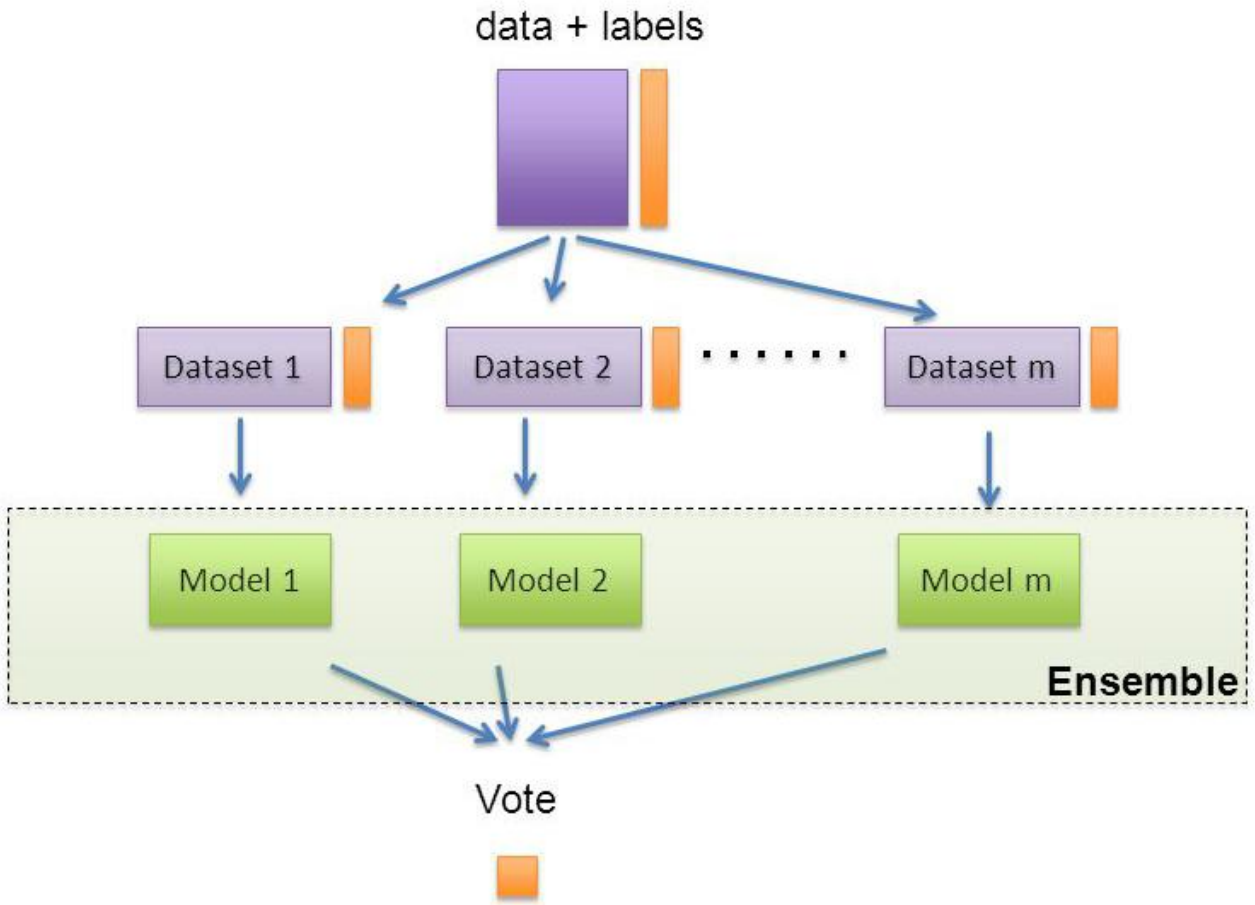
Display top 20 leaders.

Rank	Team Name	Best Score	% Improvement	Last Submit Time
1	BellKor's Pragmatic Chaos	0.8558	10.05	2009-06-26 18:42:37
Grand Prize - RMSE <= 0.8563				
2	PragmaticTheory	0.8582	9.80	2009-06-25 22:15:51
3	BellKor in BigChaos	0.8590	9.71	2009-05-13 08:14:09
4	Grand Prize Team	0.8593	9.68	2009-06-12 08:20:24
5	Dace	0.8604	9.56	2009-04-22 05:57:03
6	BigChaos	0.8613	9.47	2009-06-23 23:06:52

Netflix prize winners

- Punchline

Netflix prize conclusion: ensemble of simple methods beats one complex method



Bagging

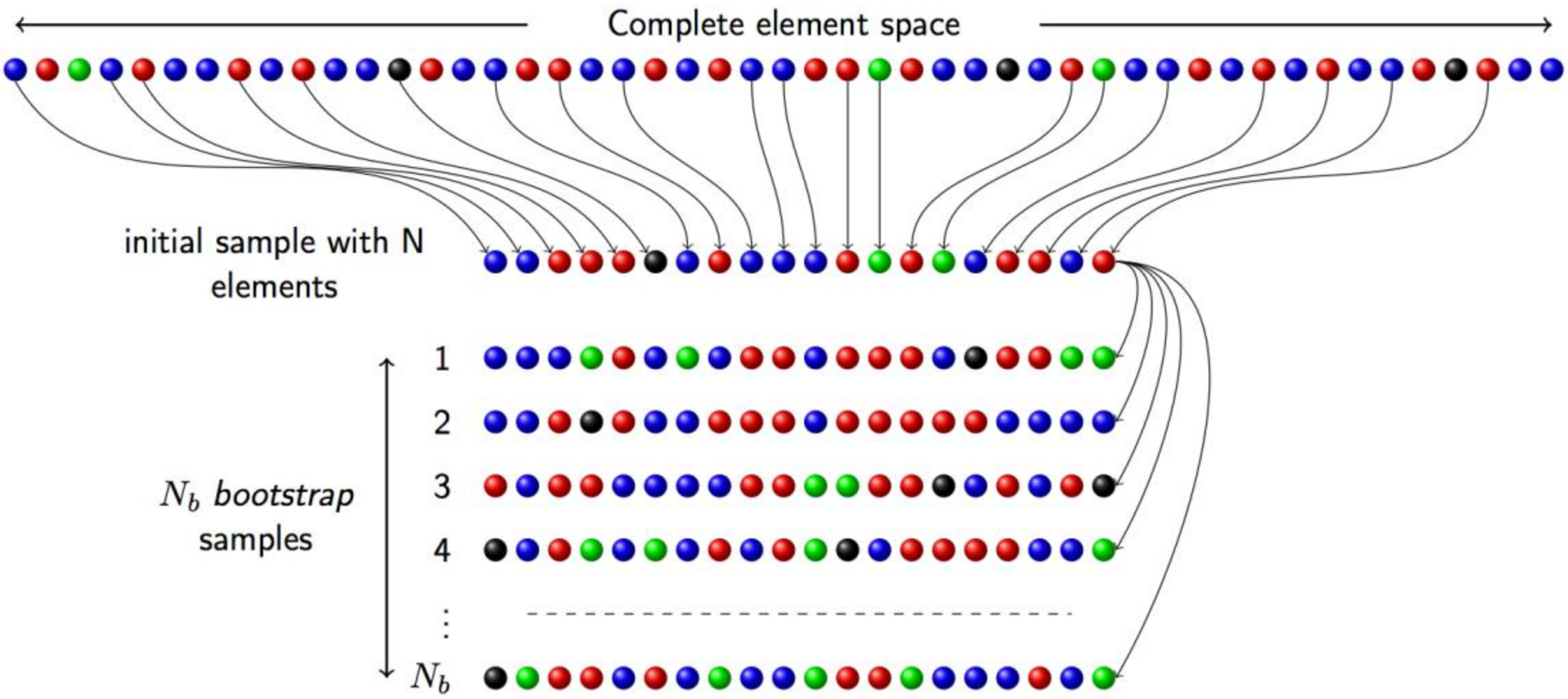
- Bagging is short for bootstrap aggregation
- **It's a general purpose method for reducing variance in any machine learning method**

- With n independent observations, $z_1, z_2, \ldots, z_n$ each with variance σ^2 , the variance of the mean ($\bar{z}$) is given by $\sigma^2/\bar{z}$
- We usually cannot do this because we don't have multiple training datasets

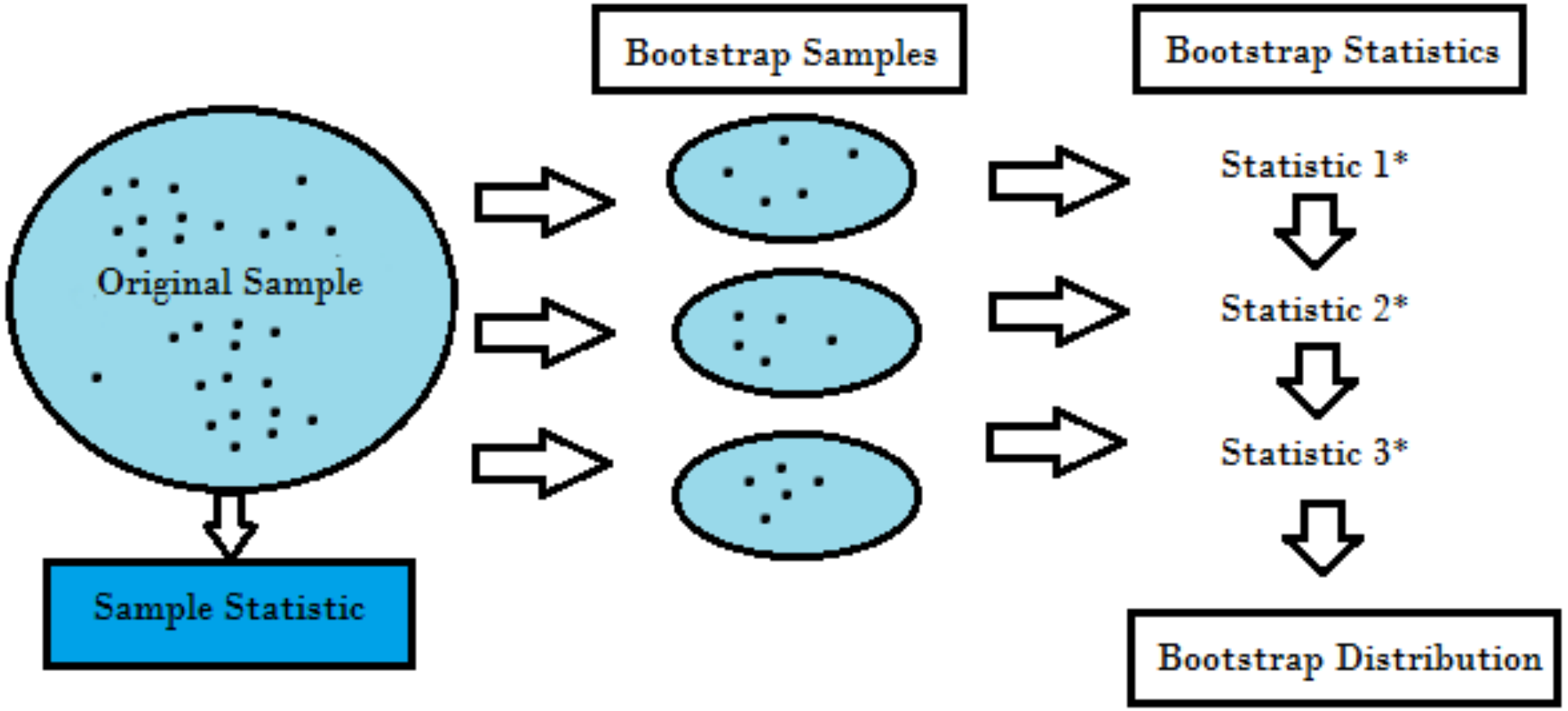
What is bootstrapping?

- In short, it's repeatedly sampling from the training dataset to develop a series of new datasets.
- E.g. if training dataset has N observations

Why bootstrapping?



What do we do with bootstrapped samples?



Bagging

- Instead, we can take repeated bootstrap samples from the training set that achieve a similar result

Bagging algorithm

1. Generate B bootstrap training datasets
2. Train method on the b-th bootstrapped set to obtain $\hat{f}^*(x)$ the prediction for data point x
3. Average all predictions to obtain average prediction over the bootstrapped samples

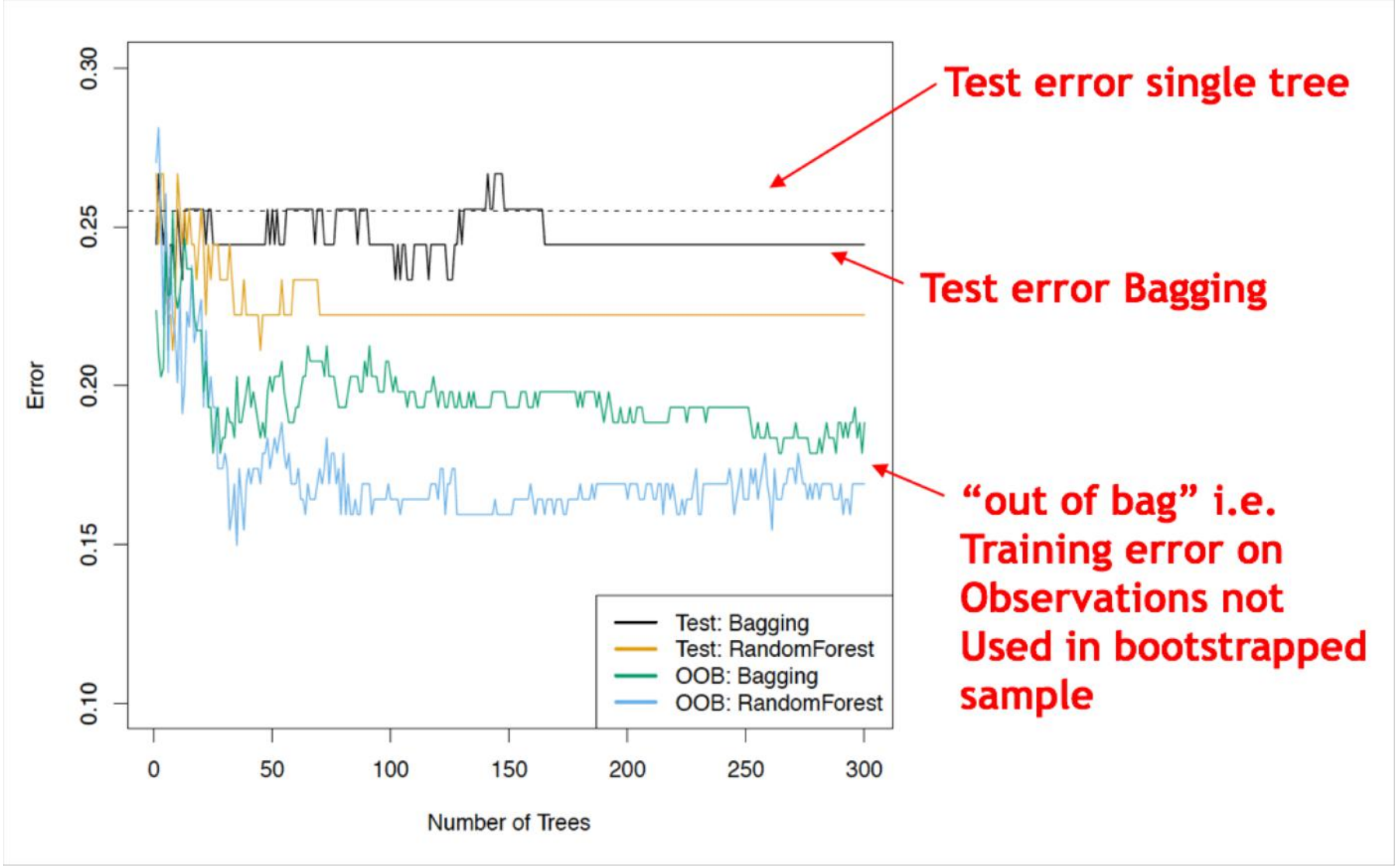
$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*\{b\}}(x)$$

South African Heart Attack Data

https://web.stanford.edu/~hastie/ElemStatLearn/datasets/SAheart.info.txt

```
A retrospective sample of males in a heart-disease high-risk region
of the Western Cape, South Africa. There are roughly two controls per
case of CHD. Many of the CHD positive men have undergone blood
pressure reduction treatment and other programs to reduce their risk
factors after their CHD event. In some cases the measurements were
made after these treatments. These data are taken from a larger
dataset, described in  Rousseauw et al, 1983, South African Medical
Journal.

sbp             systolic blood pressure
tobacco         cumulative tobacco (kg)
ldl             low densiity lipoprotein cholesterol
adiposity
famhist         family history of heart disease (Present, Absent)
typea           type-A behavior
obesity
alcohol         current alcohol consumption
age             age at onset
chd             response, coronary heart disease

To read into R:
read.table("http://www-stat.stanford.edu/~tibs/ElemStatLearn/datasets/SAheart.data",
        sep=",",head=T,row.names=1)
```

Bagging result on heart attack data



Out-of-bag error

- Recall that for each bootstrapped sample *b* is composed of a subset of the total training data
- For each sample, the data not used to fit the model is referred to as **out-of-bag (OOB) observations**
- We can better approximate out of sample error by only using out-of-bag observations for model validation

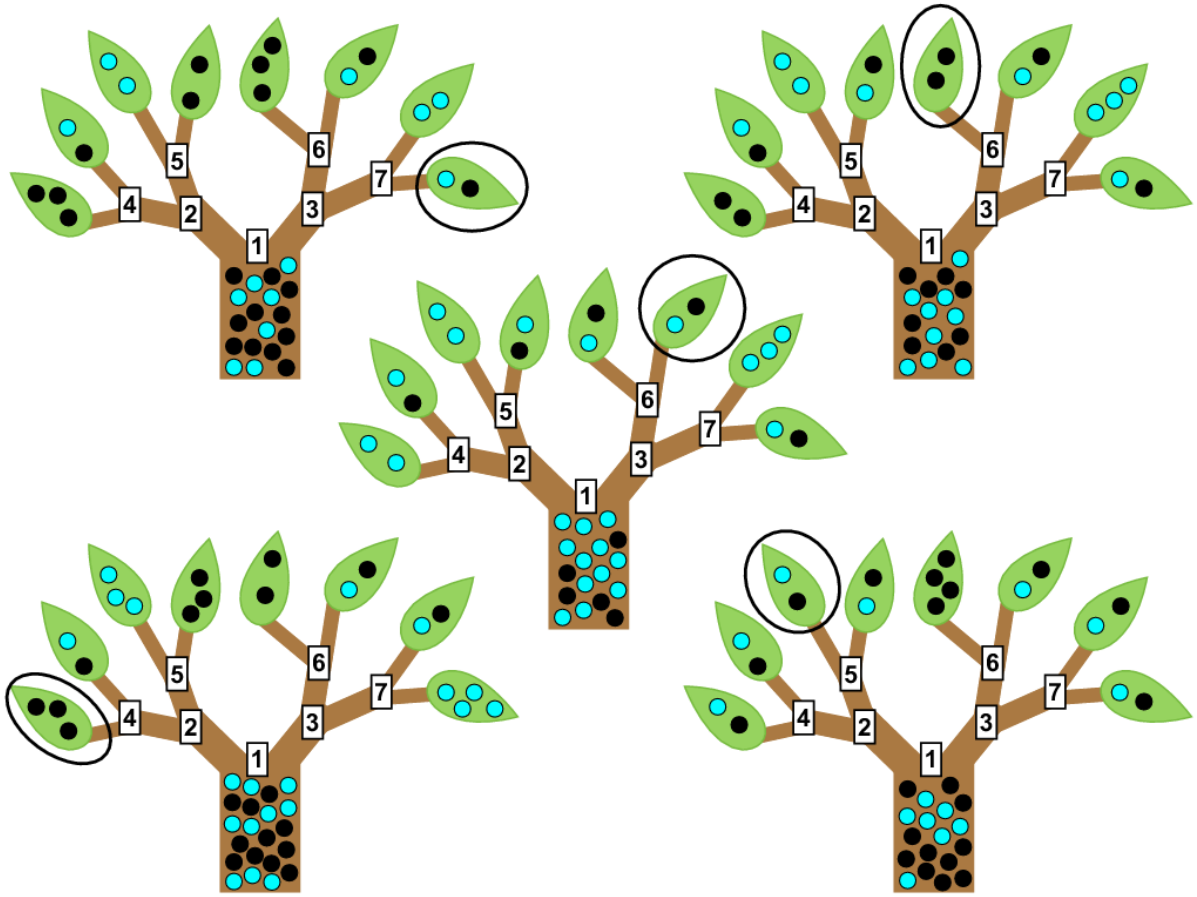
Random Forests

- Random forests are a slight trick to bagging that highly improves predictive power

- **Many trees do poorly because the stepwise greedy algorithm doesn't fully explore variable and parameter space**

Random Forests

- Random forests is like bagging, only each time a split in a tree is considered, a random selection of m predictors is chosen as split candidates
- A fresh set of m predictors is taken at each split.
- Typically we choose $m \approx \sqrt{p}$ *<u>although this is a tuning parameter!</u>*

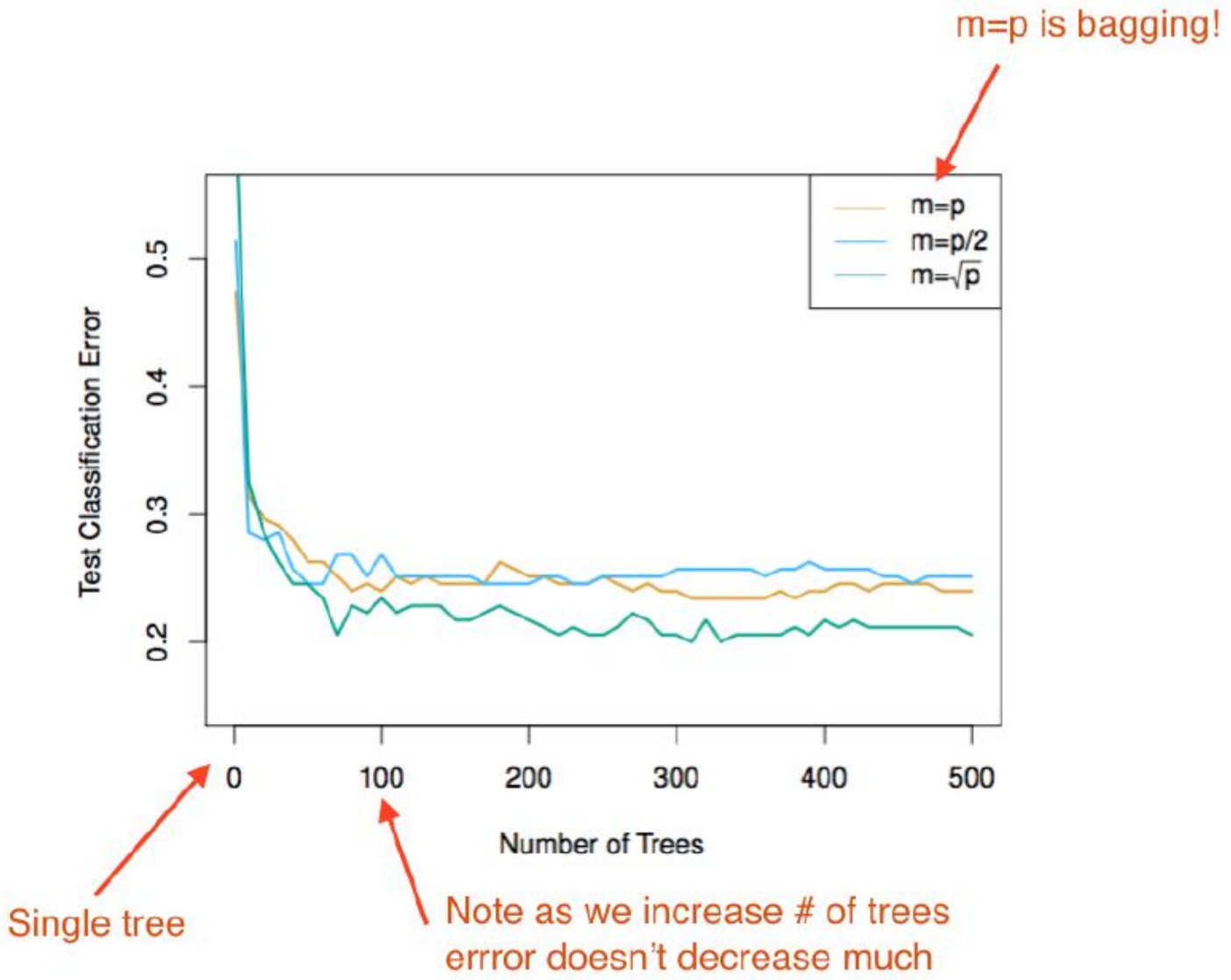


Cancer dataset

https://web.stanford.edu/~hastie/ElemStatLearn/datasets/14cancer.info

```
14-cancer  gene expression data. 16,063 genes, 144 training samples,
54 test samples.

One gene per row, one sample per column

Cancer classes are labelled as follows:

1.  breast
2.  prostate
3.  lung
4.  collerectal
5.  lymphoma
6.  bladder
7.  melanoma
8.  uterus
9.  leukemia
10. renal
11. pancreas
12. ovary
13. meso
14. cns

Reference:

S. Ramaswamy and P.  Tamayo and  R. Rifkin and S. Mukherjee and C.H. Yeang and
M. Angelo and C. Ladd and M. Reich and E. Latulippe and J.P. Mesirov and
T. Poggio and W. Gerald and M. Loda and E.S. Lander and  T.R. Golub (2001)

Multiclass Cancer Diagnosis Using Tumor Gene Expression Signatures

Proc. Natl. Acad. Sci., 98, p15149-15154.
```

Example random forest: cancer data



Estimating Random Forest Models Using "randomForest"

randomForest {randomForest} R Documentat

Classification and Regression with Random Forest

Description

randomForest implements Breiman's random forest algorithm (based on Breiman and Cutler's original Fortran code) for classification an regression. It can also be used in unsupervised mode for assessing proximities among data points.

Usage

```
## S3 method for class 'formula'
randomForest(formula, data=NULL, ..., subset, na.action=na.fail)
## Default S3 method:
randomForest(x, y=NULL,  xtest=NULL, ytest=NULL, ntree=500,
             mtry=if (!is.null(y) && !is.factor(y))
             max(floor(ncol(x)/3), 1) else floor(sqrt(ncol(x))),
             replace=TRUE, classwt=NULL, cutoff, strata,
             sampsize = if (replace) nrow(x) else ceiling(.632*nrow(x)),
             nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,
             maxnodes = NULL,
             importance=FALSE, localImp=FALSE, nPerm=1,
             proximity, oob.prox=proximity,
             norm.votes=TRUE, do.trace=FALSE,
             keep.forest=!is.null(y) && is.null(xtest), corr.bias=FALSE,
             keep.inbag=FALSE, ...)
## S3 method for class 'randomForest'
print(x, ...)
```

Arguments

data	an optional data frame containing the variables in the model. By default the variables are taken from the environment which randomForest is called from.
subset	an index vector indicating which rows should be used. (NOTE: If given, this argument must be named.)
na.action	A function to specify the action to be taken if NAs are found. (NOTE: If given, this argument must be named.)
x, formula	a data frame or a matrix of predictors, or a formula describing the model to be fitted (for the print method, an randomForest object).
y	A response vector. If a factor, classification is assumed, otherwise regression is assumed. If omitted, randomForest will run in unsupervised mode.
xtest	a data frame or matrix (like x) containing predictors for the test set.
ytest	response for the test set.
ntree	Number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times.
mtry	Number of variables randomly sampled as candidates at each split. Note that the default values are different for classification (sqrt(p) where p is number of variables in x) and regression (p/3)
replace	Should sampling of cases be done with or without replacement?
classwt	Priors of the classes. Need not add up to one. Ignored for regression.
cutoff	(Classification only) A vector of length equal to number of classes. The 'winning' class for an observation is the one with the maximum ratio of proportion of votes to cutoff. Default is 1/k where k is the number of classes (i.e., majority vote wins).
strata	A (factor) variable that is used for stratified sampling.

Estimating a Random Forest Model Against the Boston Data

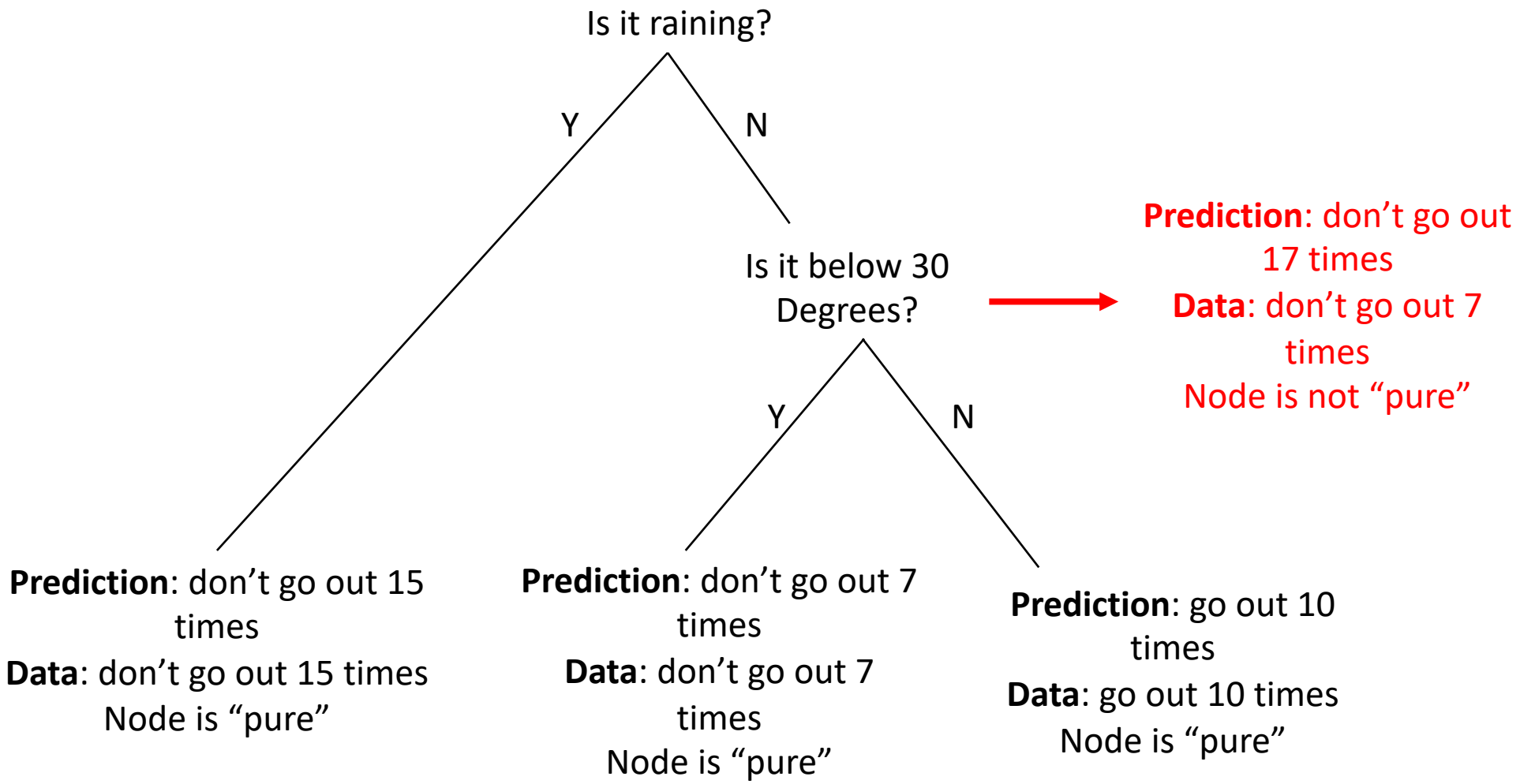
```
require('randomForest')
require('MASS')
set.seed(101)
dim(Boston)

#set a training set
train = sample(1:nrow(Boston),300)

#fit a random forest and see how well it performs
rf.boston = randomForest(medv~.,data=Boston,subset=train)
rf.boston
```

```
> rf.boston = randomForest(medv~.,data=Boston,subset=train)
> rf.boston

Call:
 randomForest(formula = medv ~ ., data = Boston, subset = train)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 4

          Mean of squared residuals: 12.30718
                    % Var explained: 85.13
```

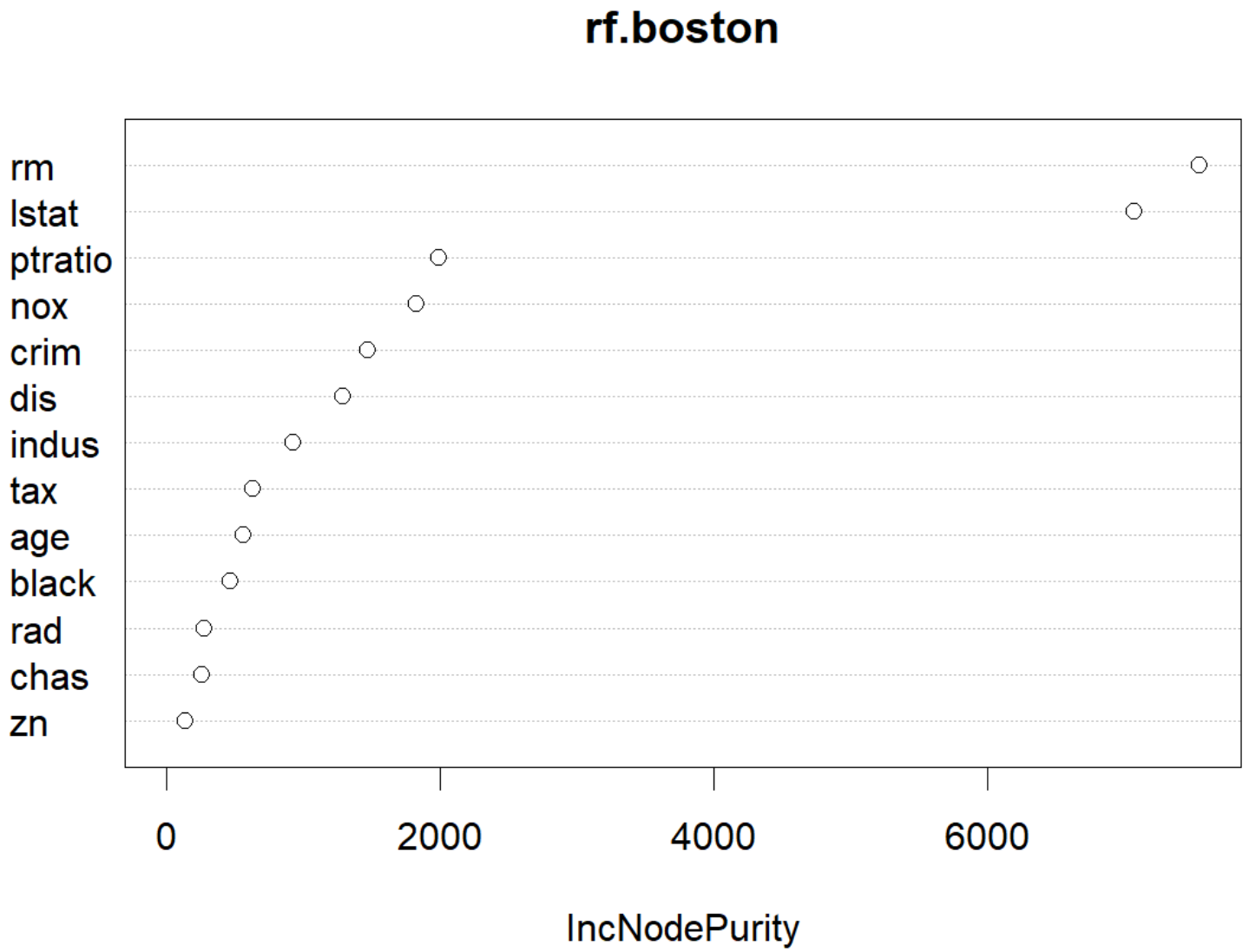
Variable “Importance”

```
> importance(rf.boston)
        IncNodePurity
crim        1476.8398
zn           138.0405
indus        930.0199
chas         265.7788
nox         1831.5730
rm          7551.2292
age          567.3393
dis         1296.5395
rad          280.9853
tax          639.1153
ptratio     1990.4504
black        469.8464
lstat       7076.4031
```

What is node “purity”?



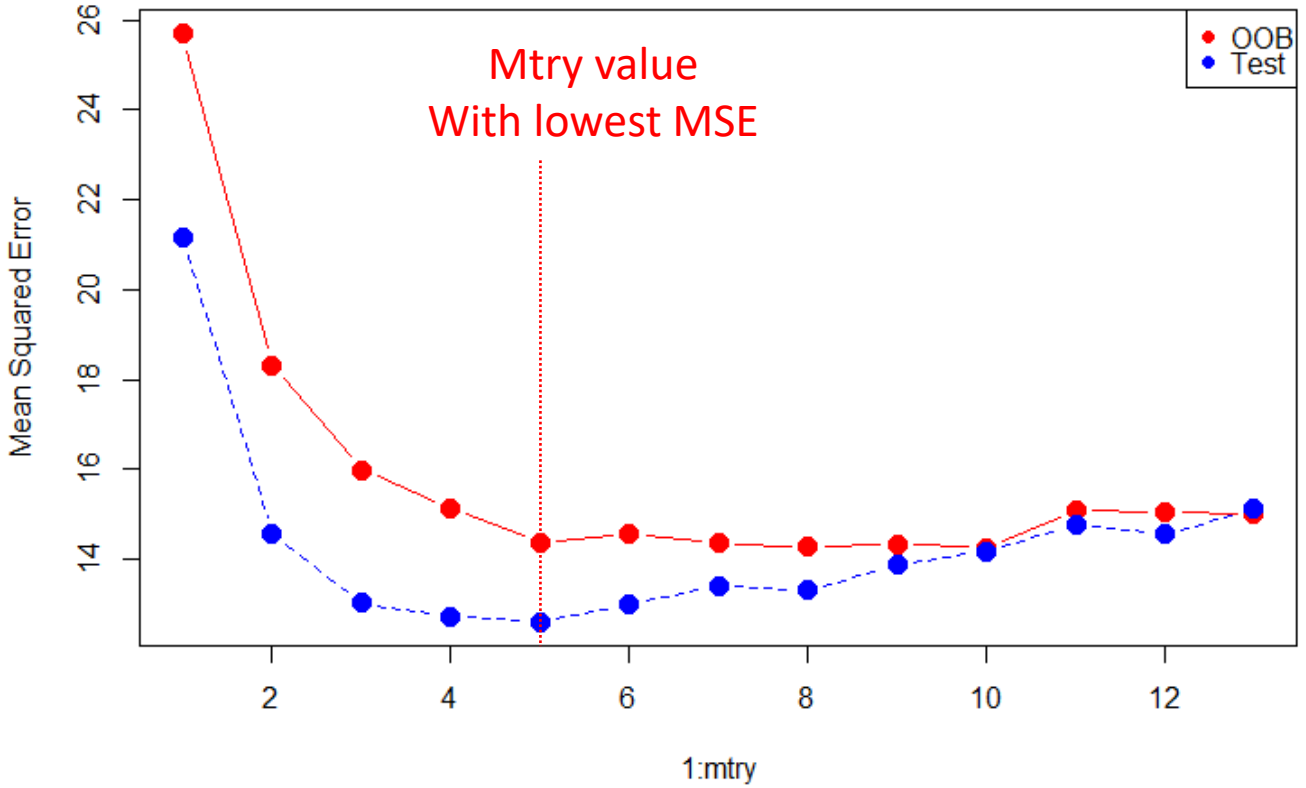
Plotting Variable “Importance”



Cross-validate to select “mtry”

```
oob.err=double(13)
test.err=double(13)
for(mtry in 1:13){
  # the the RF with that value of mtry, number of vars chosen at each split
  fit=randomForest(medv~.,data=Boston,subset=train,mtry=mtry,ntree=400)
  oob.err[mtry]=fit$mse[400]
  pred=predict(fit,Boston[-train,])
  test.err[mtry]=with(Boston[-train,],mean((medv-pred)^2))
  cat(mtry," ")
}
```

Cross-validate to select “mtry”




```
# matplot(x,y) plots the columns of one matrix against the columns of another
matplot(1:mtry,cbind(test.err,oob.err),pch=19,
        col=c("red","blue"),type="b",ylab="Mean Squared Error")
legend("topright",legend=c("OOB","Test"),pch=19,col=c("red","blue"))
```