Class 9

BUS 696

Prof. Jonathan Hersh

BUS 696: Class 9 Announcements

1. Pset 7 solutions
2. Problem Set 8 Posted
3. Final Project
    - One sheet due today
4. Reminder: midterm exam next week
    - 4-5 hours to do exam
5. Any other Qs?

BUS 696: Class 9 Outline

1. AI Would You Invest?
2. Regression Trees
3. Regression Trees in R
4. Example Test Q

AI-POWERED FOOD WASTE FIRM WINNOW SECURES $20M

The technology allows restaurants including those in IKEA to automatically track and then reduce wasted food

BC Artificial Intelligen...

Alistair Hardaker 11:49am 17th Oct 2019

The Winnow hardware

Winnow, a tech firm using AI to cut food waste, has secured a total of $20m in the last month.

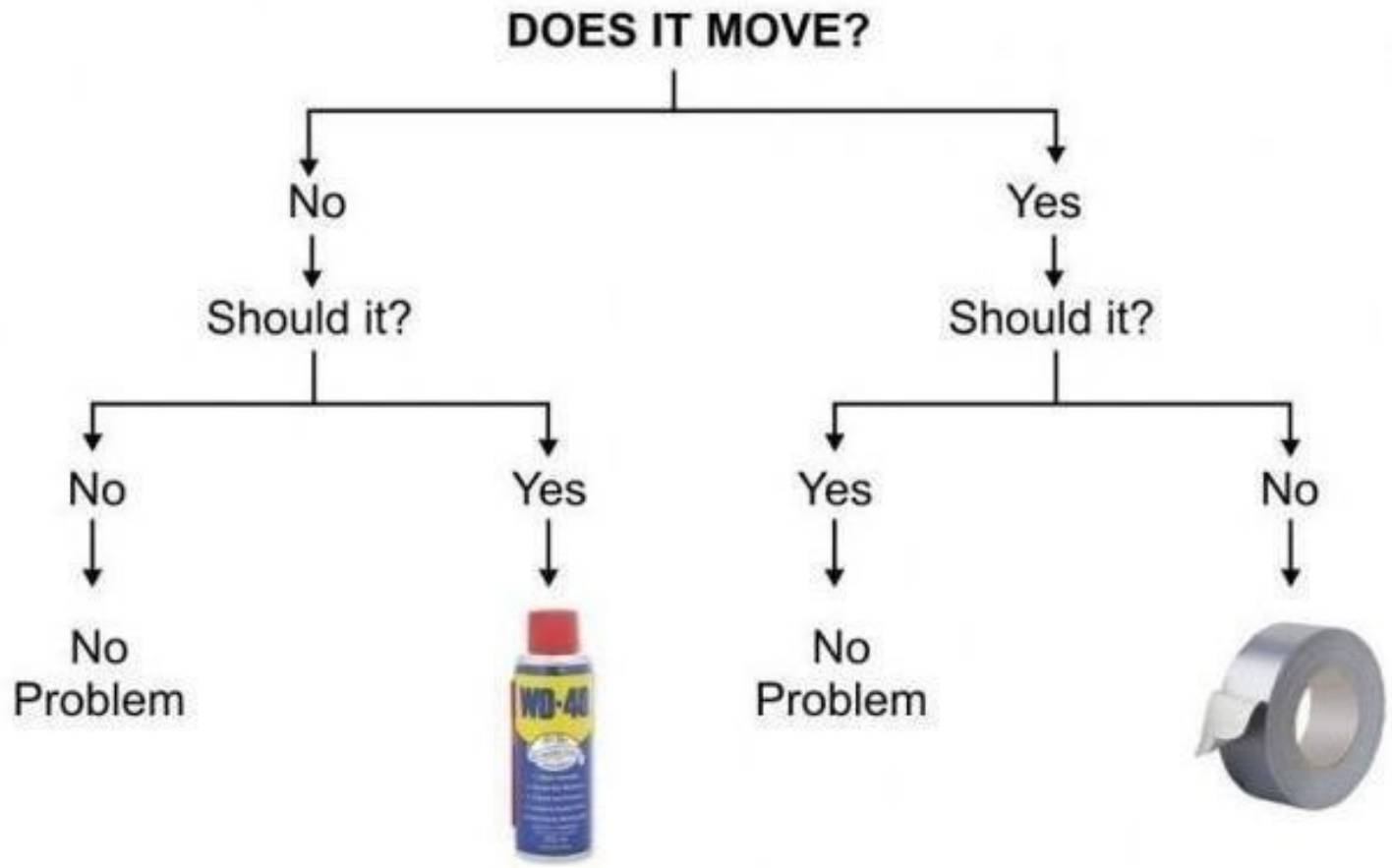
The funding comes in the form of a $12m Series B funding round led by Ingka Group and Mustard Seed and an $8m loan from The European Investment Bank (EIB).

The London firm offers Winnow Vision, AI powered technology to help chefs automate waste tracking

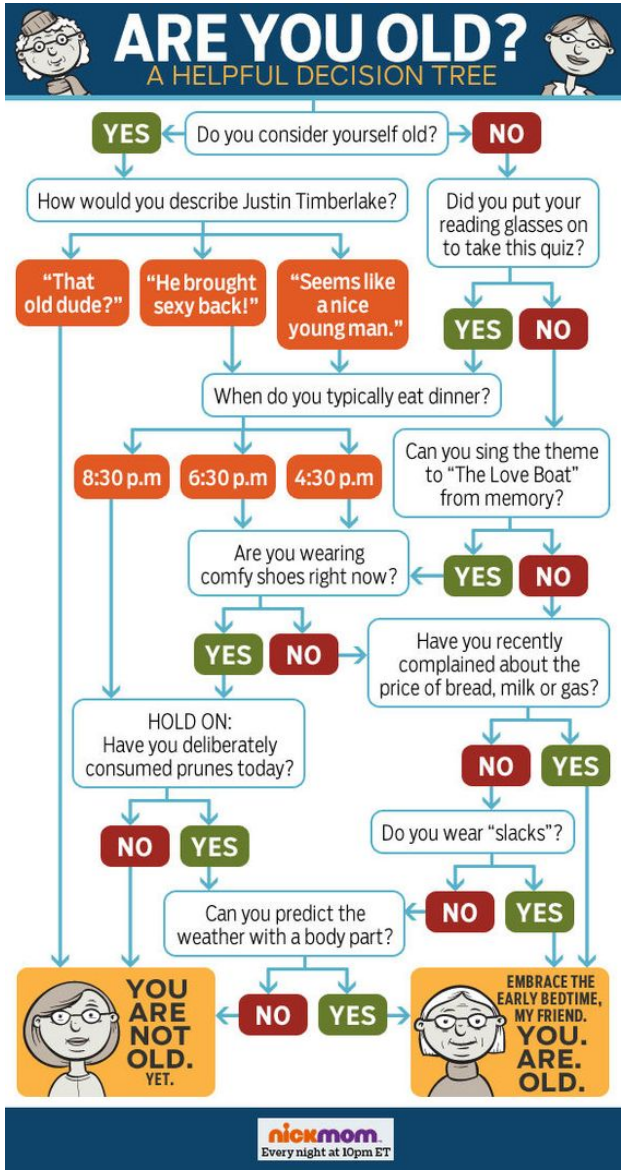
The tech works by taking photos of wasted food as it's thrown away and uses the images to recognise what has been discarded.

The systems have reportedly surpassed human levels of accuracy in identifying wasted foods allowing kitchens to automatically register food waste without any human interaction.

Binary Decision Rules

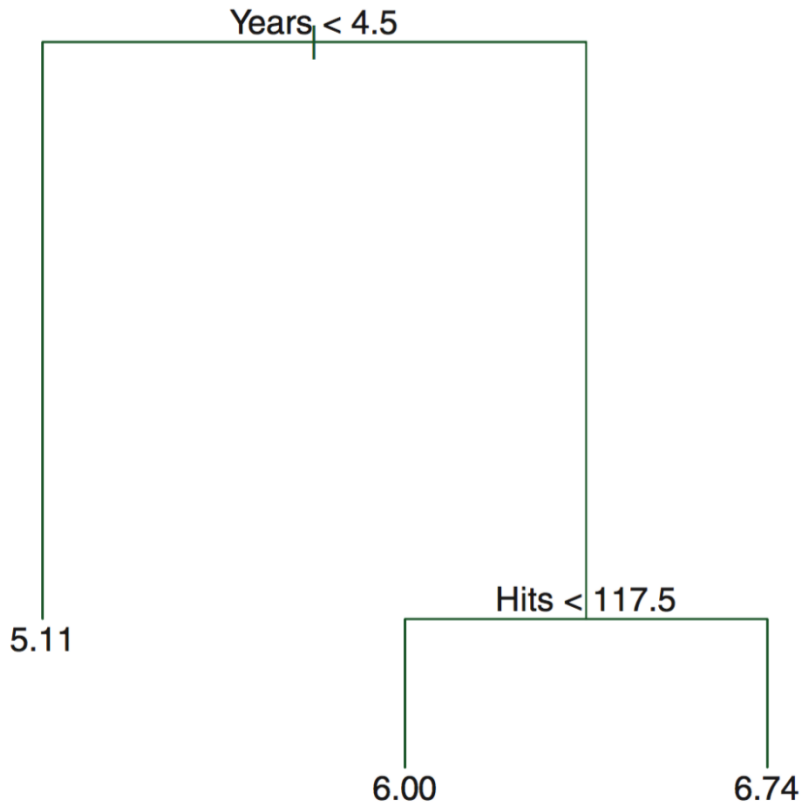


Classification Trees: Series of Binary Decision



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

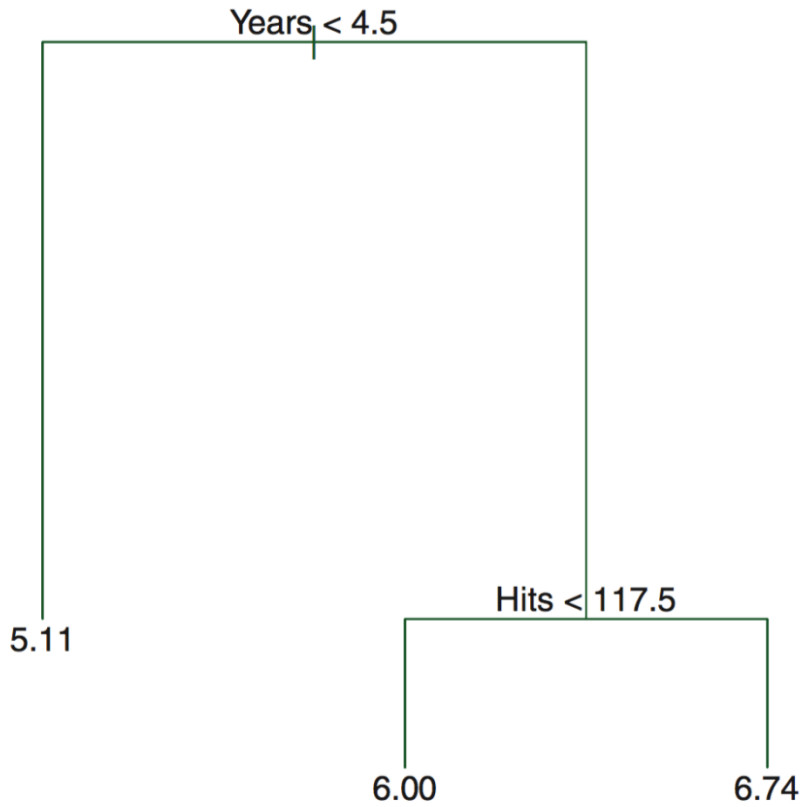
- Simple
- Easy to interpret
- Easy to explain
- Can be displayed graphically!
- Bagging, boosting, and random forests very powerful (combining trees)

Cons

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

Decision/Regression Trees

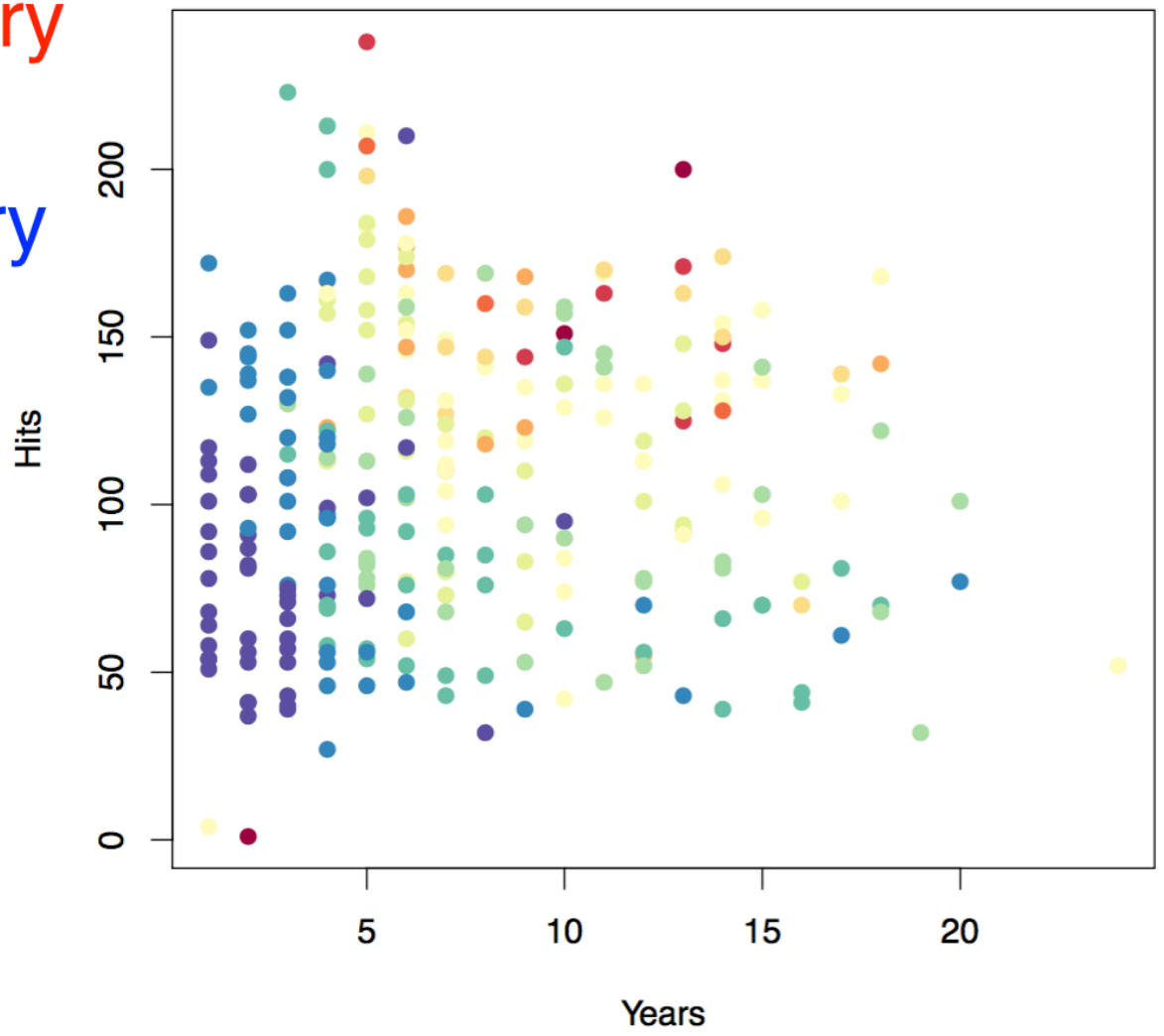
- Decision trees can be applied to both regression problems ($y_i \in R$) and classification problems $y_i \in \{class1, class2, \dots, \}$
- We'll consider both



Baseball salary data: how to partition/stratify?

High salary
red

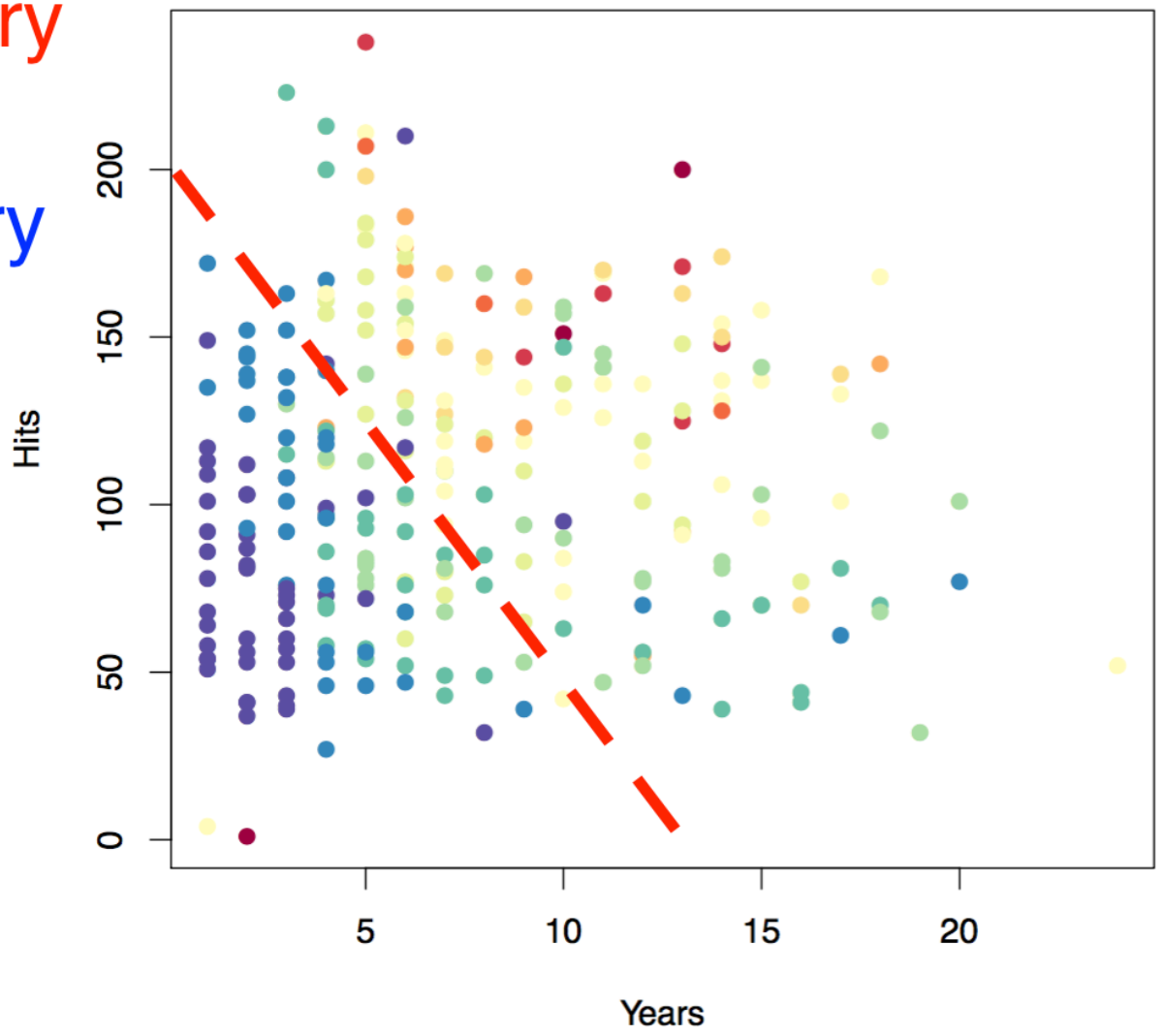
Low salary
blue



Baseball salary data: how to partition/stratify?

High salary
red

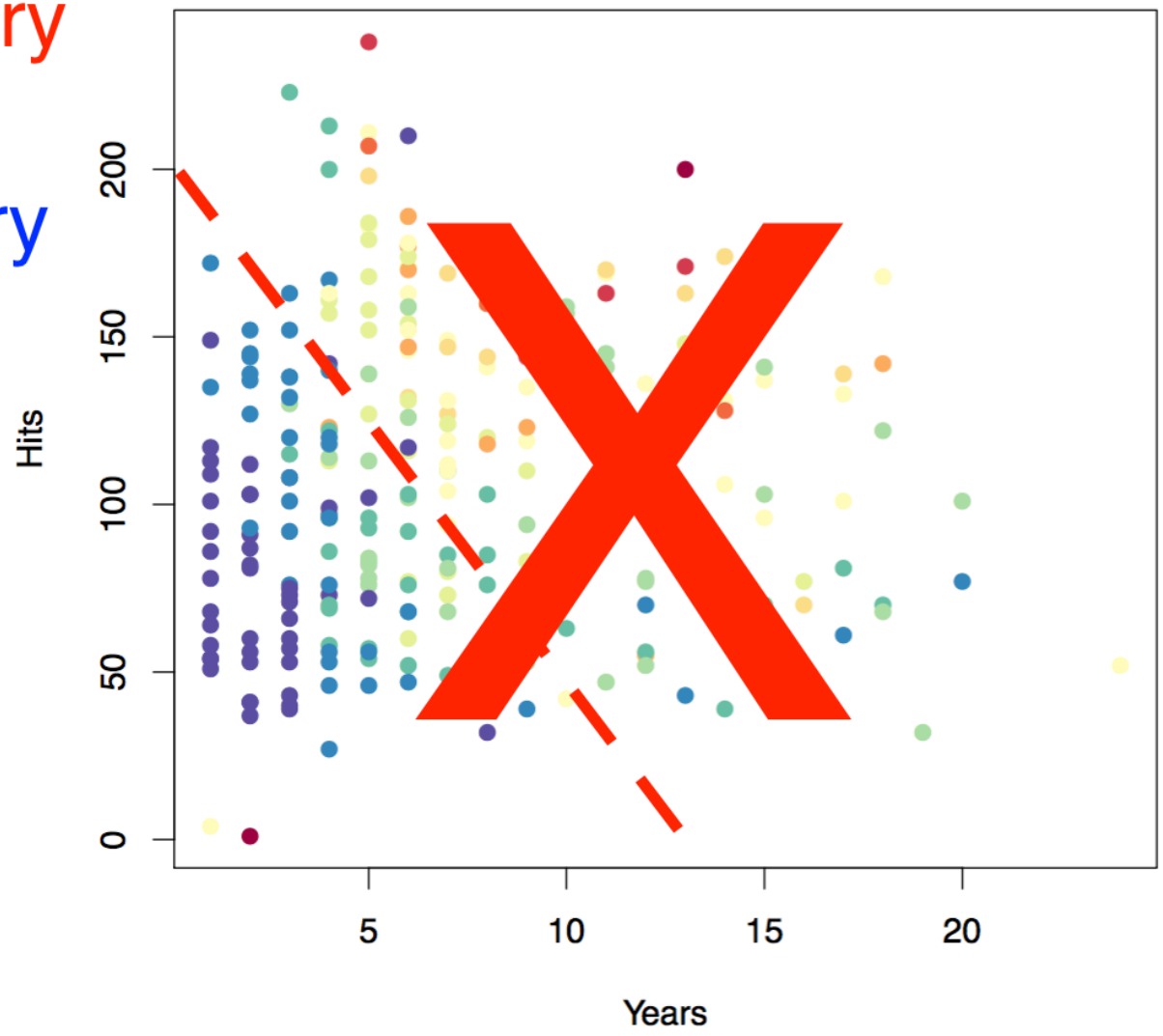
Low salary
blue



Baseball salary data: how to partition/stratify?

High salary
red

Low salary
blue

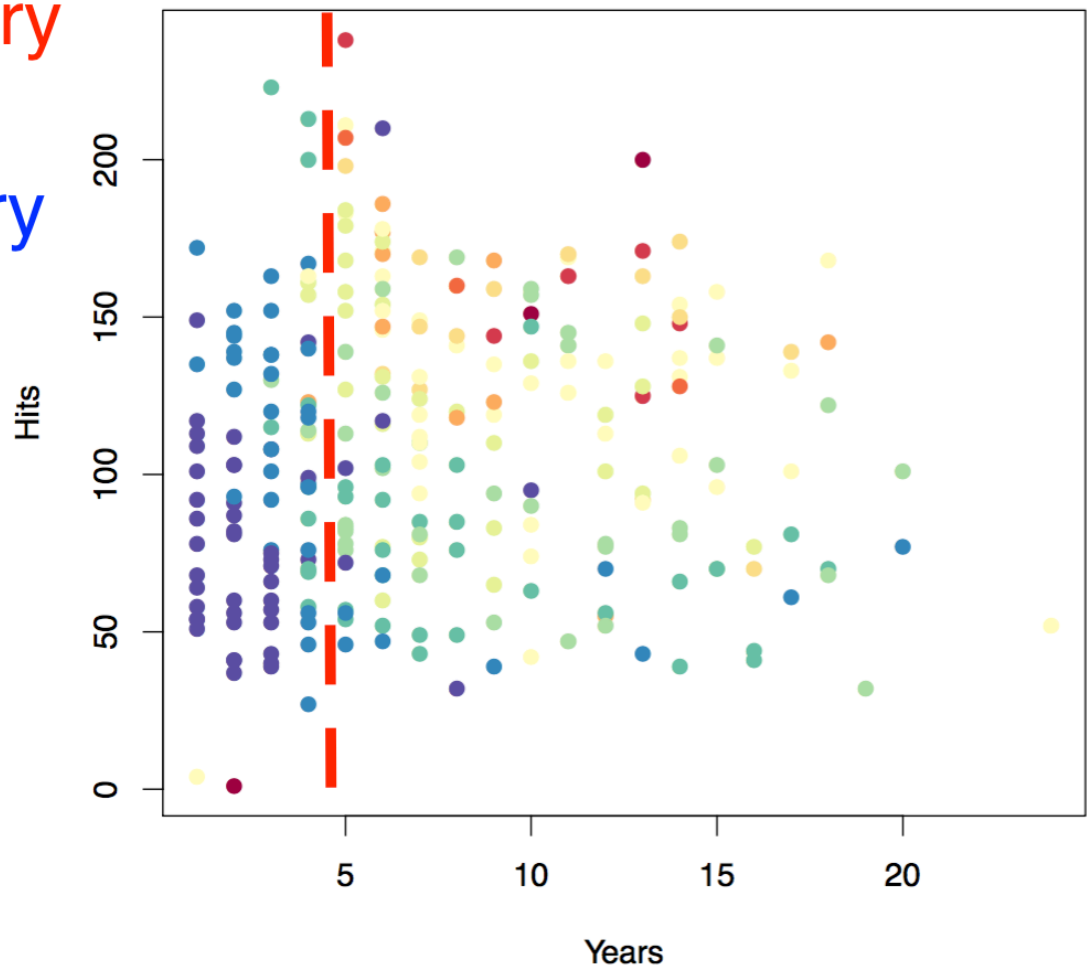


- Only linear classification rules are allowed, e.g. year > 10

Baseball salary data: split 1

High salary
red

Low salary
blue

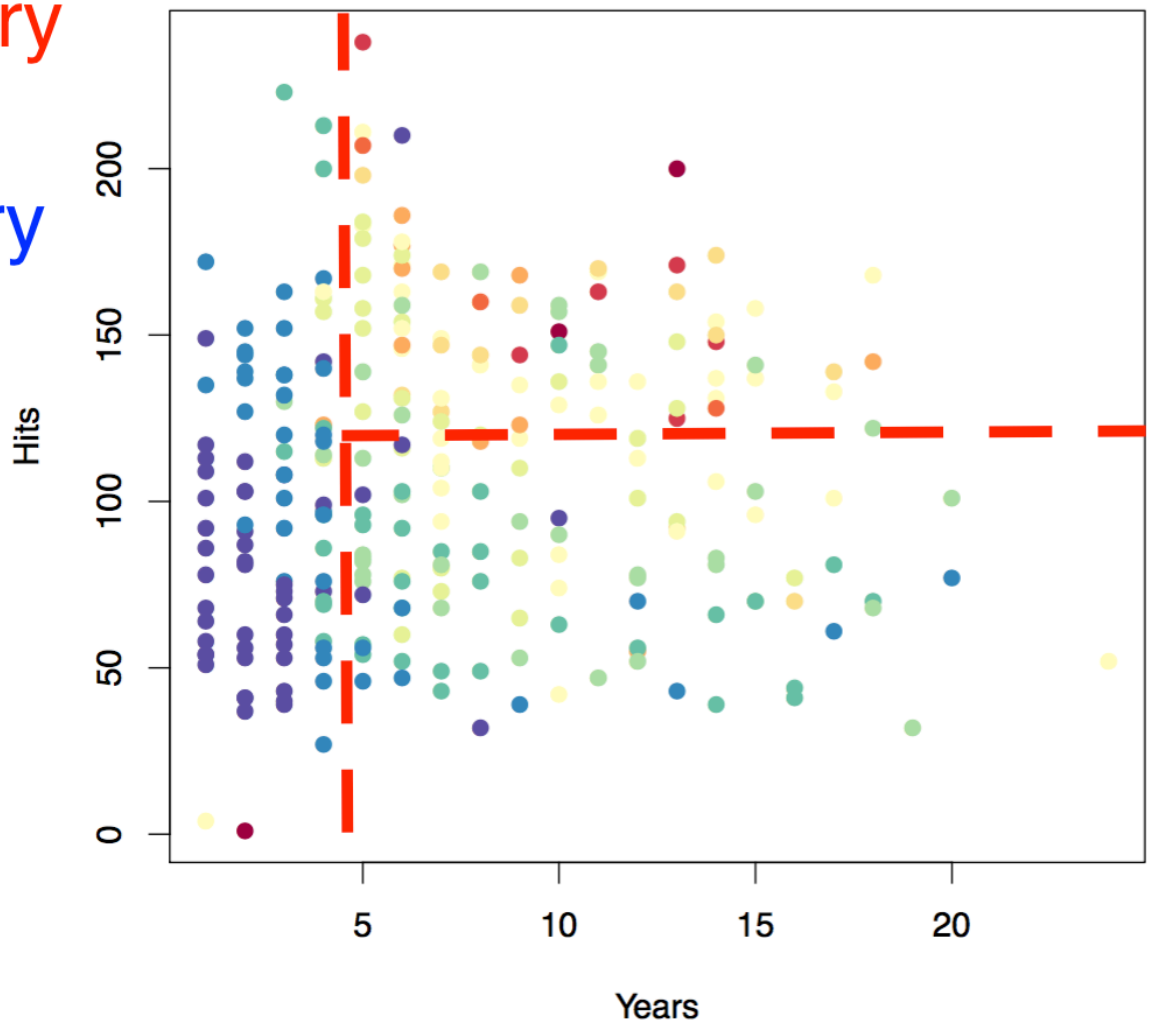


- Split 1: years > 4.5

Baseball salary data: split 2

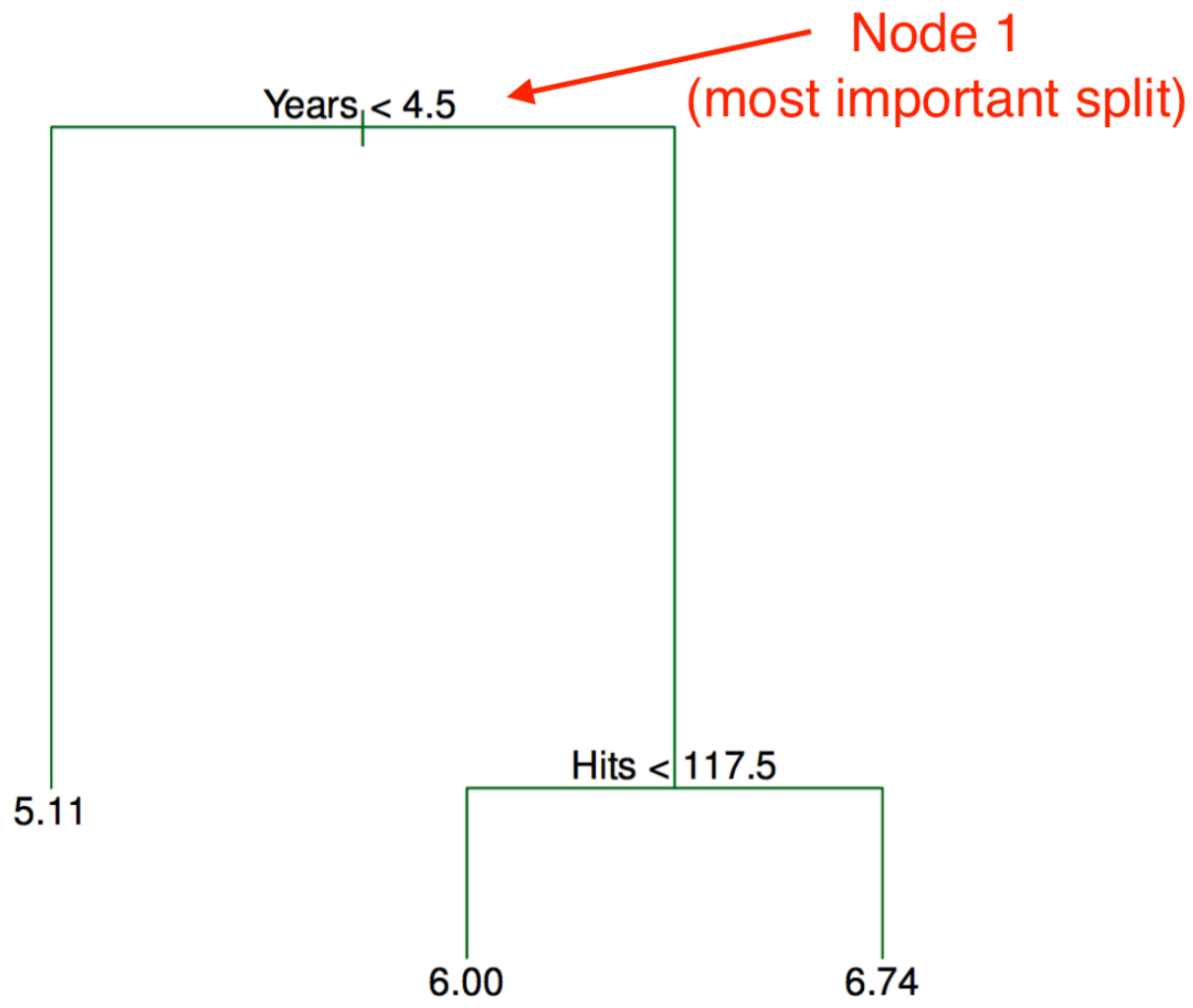
High salary
red

Low salary
blue



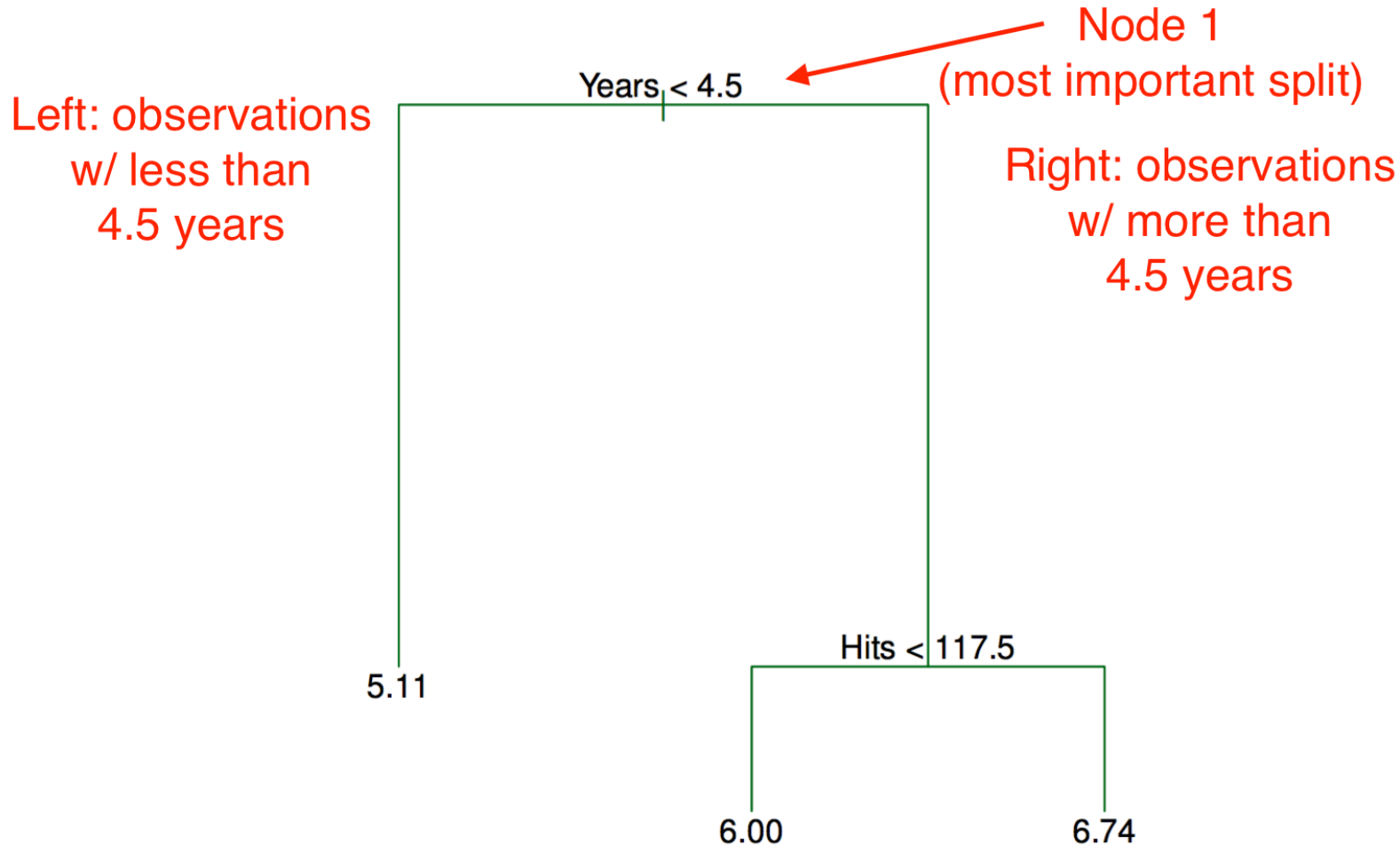
- Split 1: years > 4.5
- Split 2: hits > 117.5

Tree Representation

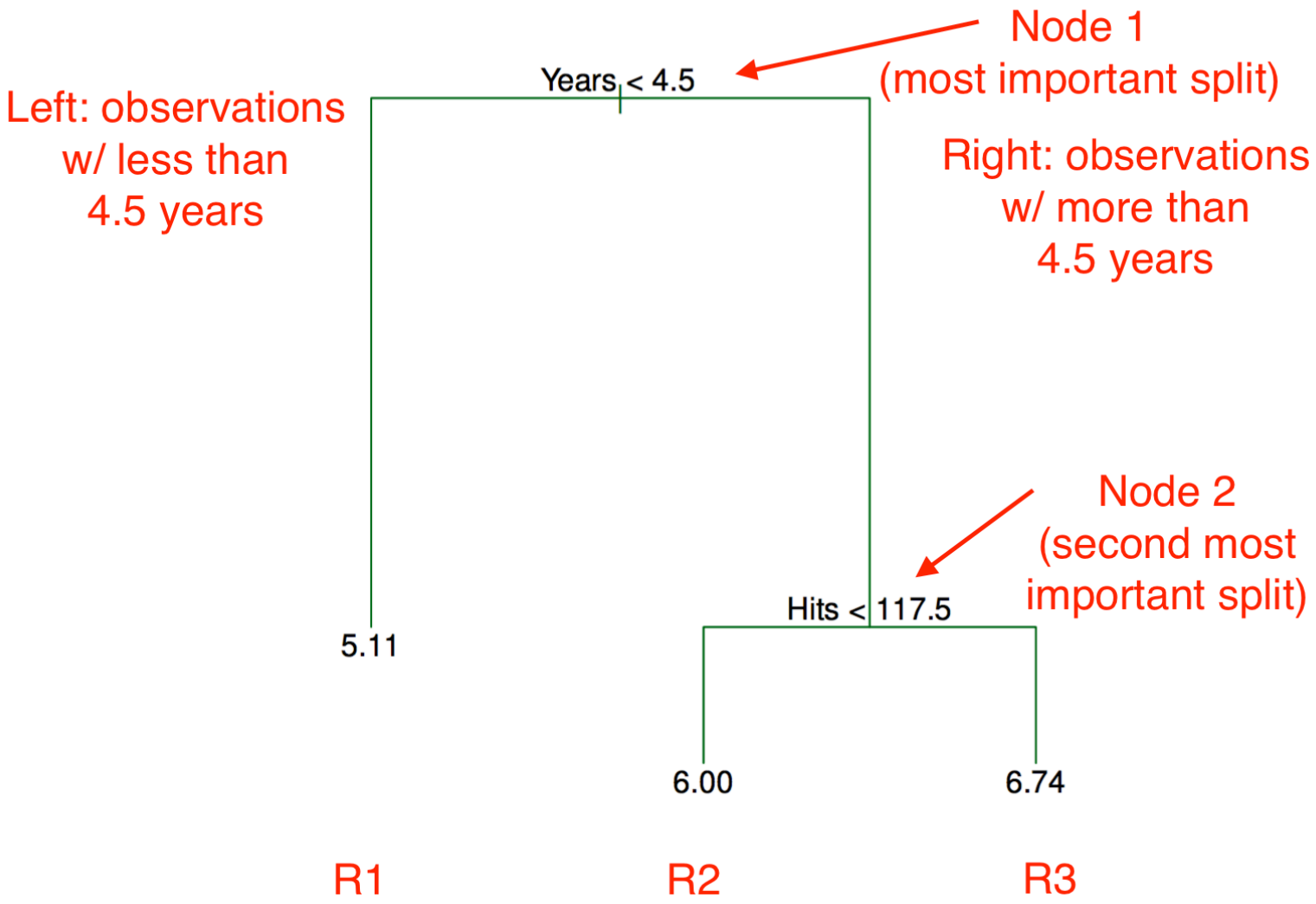


- Trees are read top-down
- Most important split is at top
- Length represents how much within-cluster variance decreases from split

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

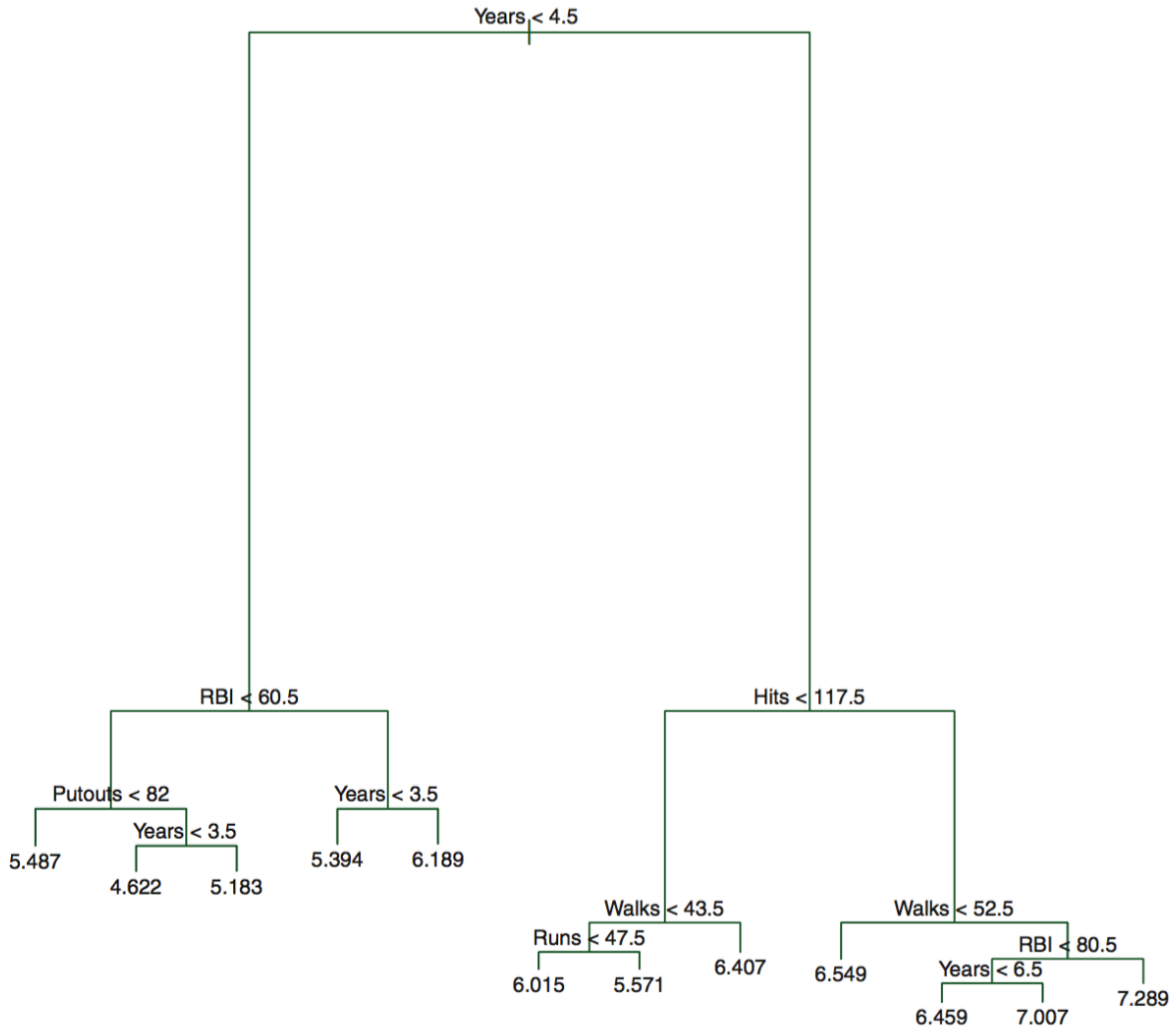


FIGURE 8.4. *Regression tree analysis for the* `Hitters` *data. The unpruned tree that results from top-down greedy splitting on the training data is shown.*

Cross-validate to find alpha

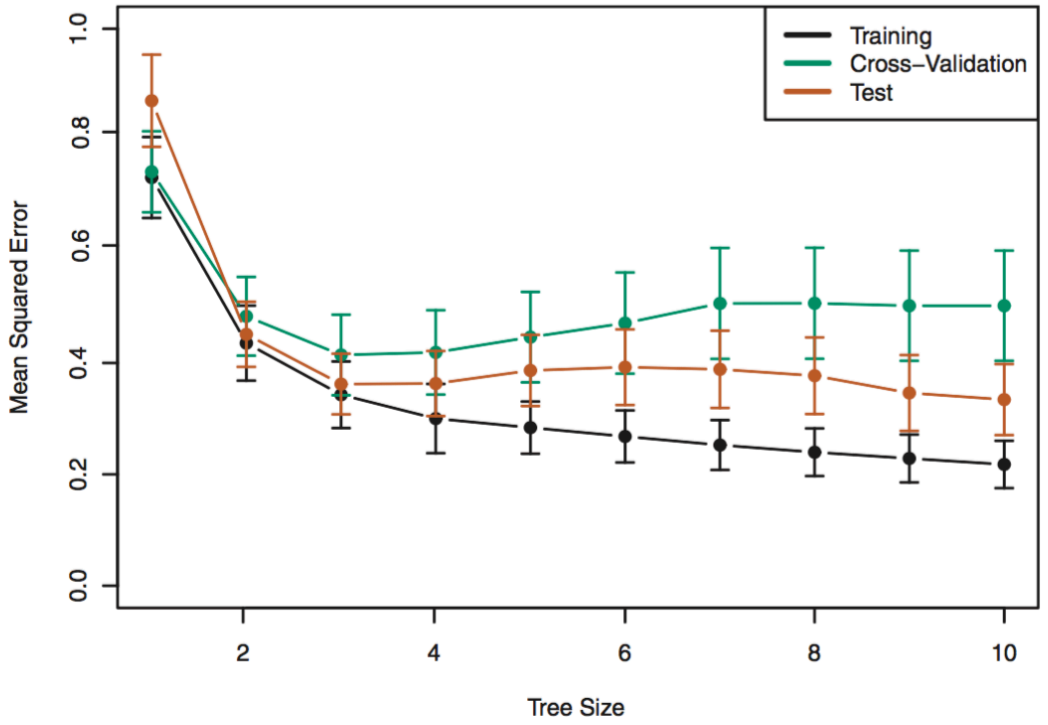
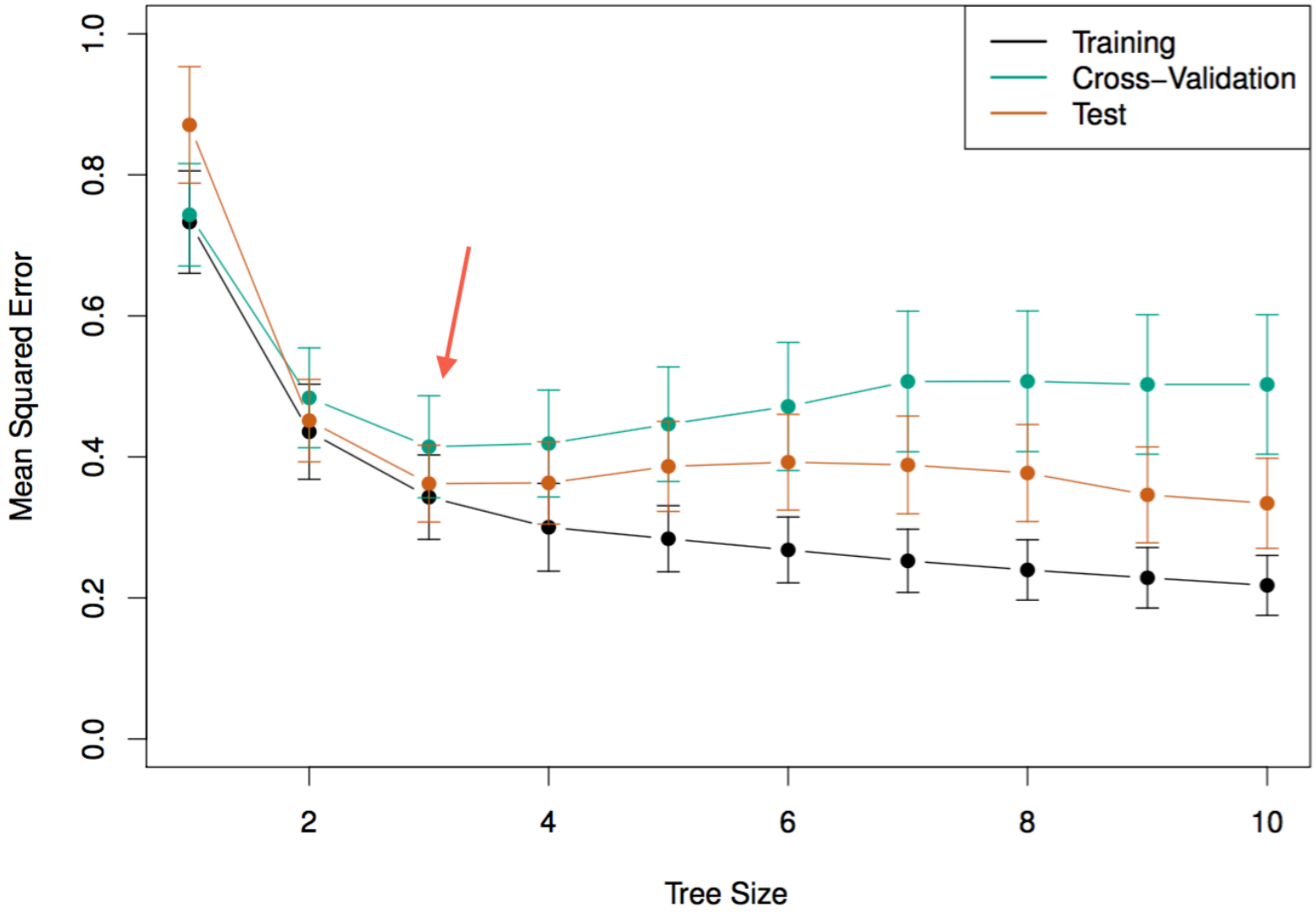


FIGURE 8.5. *Regression tree analysis for the* `Hitters` *data. The training, cross-validation, and test MSE are shown as a function of the number of terminal nodes in the pruned tree. Standard error bands are displayed. The minimum cross-validation error occurs at a tree size of three.*

Cross-validate to find alpha



Titanic dataset in ‘titanic’ package

`titanic_train` *Titanic train data.*

Description

Titanic train data.

Usage

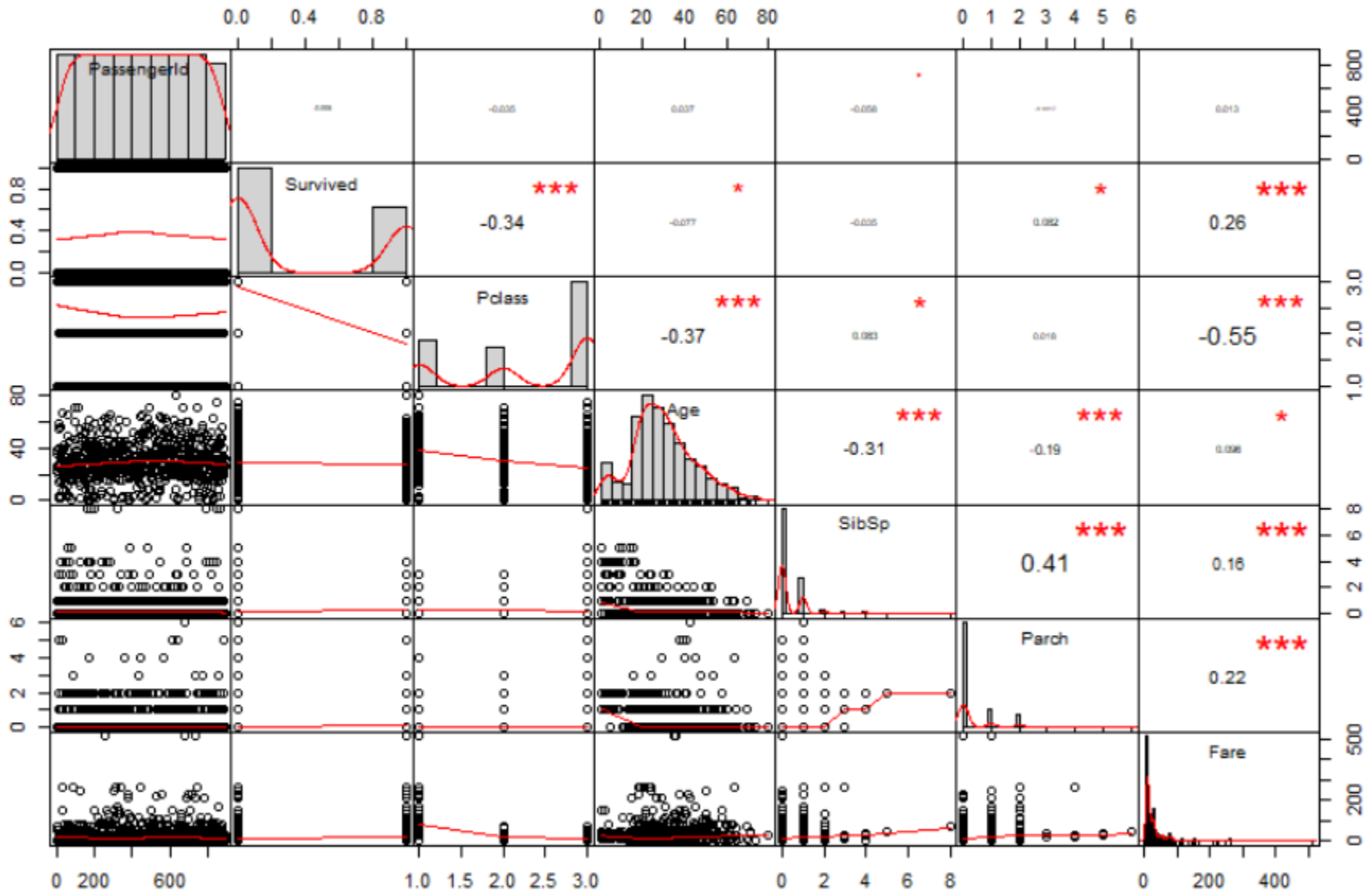
```
titanic_train
```

Format

Data frame with columns

PassengerId Passenger ID
Survived Passenger Survival Indicator
Pclass Passenger Class
Name Name
Sex Sex
Age Age
SibSp Number of Siblings/Spouses Aboard
Parch Number of Parents/Children Aboard
Ticket Ticket Number
Fare Passenger Fare
Cabin Cabin
Embarked Port of Embarkation

Titanic dataset in ‘titanic’ package



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

ctree {partykit} R Documentation

Conditional Inference Trees

Description

Recursive partitioning for continuous, censored, ordered, nominal and multivariate response variables in a conditional inference framework.

Usage

```
ctree(formula, data, subset, weights, na.action = na.pass, offset, cluster,
    control = ctree_control(...), ytrafo = NULL,
    converged = NULL, scores = NULL, doFit = TRUE, ...)
```

Arguments

`formula`	a symbolic description of the model to be fit.
`data`	a data frame containing the variables in the model.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`weights`	an optional vector of weights to be used in the fitting process. Only non-negative integer valued weights are allowed.
`offset`	an optional vector of offset values.
`cluster`	an optional factor indicating independent clusters. Highly experimental, use at your own risk.
`na.action`	a function which indicates what should happen when the data contain missing value.
`control`	a list with control parameters, see `ctree_control`.
`ytrafo`	an optional named list of functions to be applied to the response variable(s) before testing their association with the explanatory variables. Note that this transformation is only performed once for the root node and does not take weights into account. Alternatively, `ytrafo` can be a function of `data` and `weights`. In this case, the transformation is computed for every node with corresponding weights. This feature is experimental and the user interface likely to change.
`converged`	an optional function for checking user-defined criteria before splits are implemented. This is not to be used and very likely to change.
`scores`	an optional named list of scores to be attached to ordered factors.
`doFit`	a logical, if `FALSE`, the tree is not fitted.
`...`	arguments passed to `ctree_control`.

Estimate a tree model to predict titanic survival

```r
# clean data
titanic_df <- titanic_train %>% as_tibble() %>%
  mutate(Suvived = if_else(Survived == 1,
                           "Survived", "Dead"),
         Survived = as.factor(Survived),
         Sex = as.factor(Sex),
         Pclass = as.factor(Pclass)) %>%
         mutate_if(is.character, as.factor)
```

```r
# ctree to estimate model
titanic_tree <- ctree(Survived ~ Sex + Pclass,
                      data = titanic_df)
titanic_tree
```

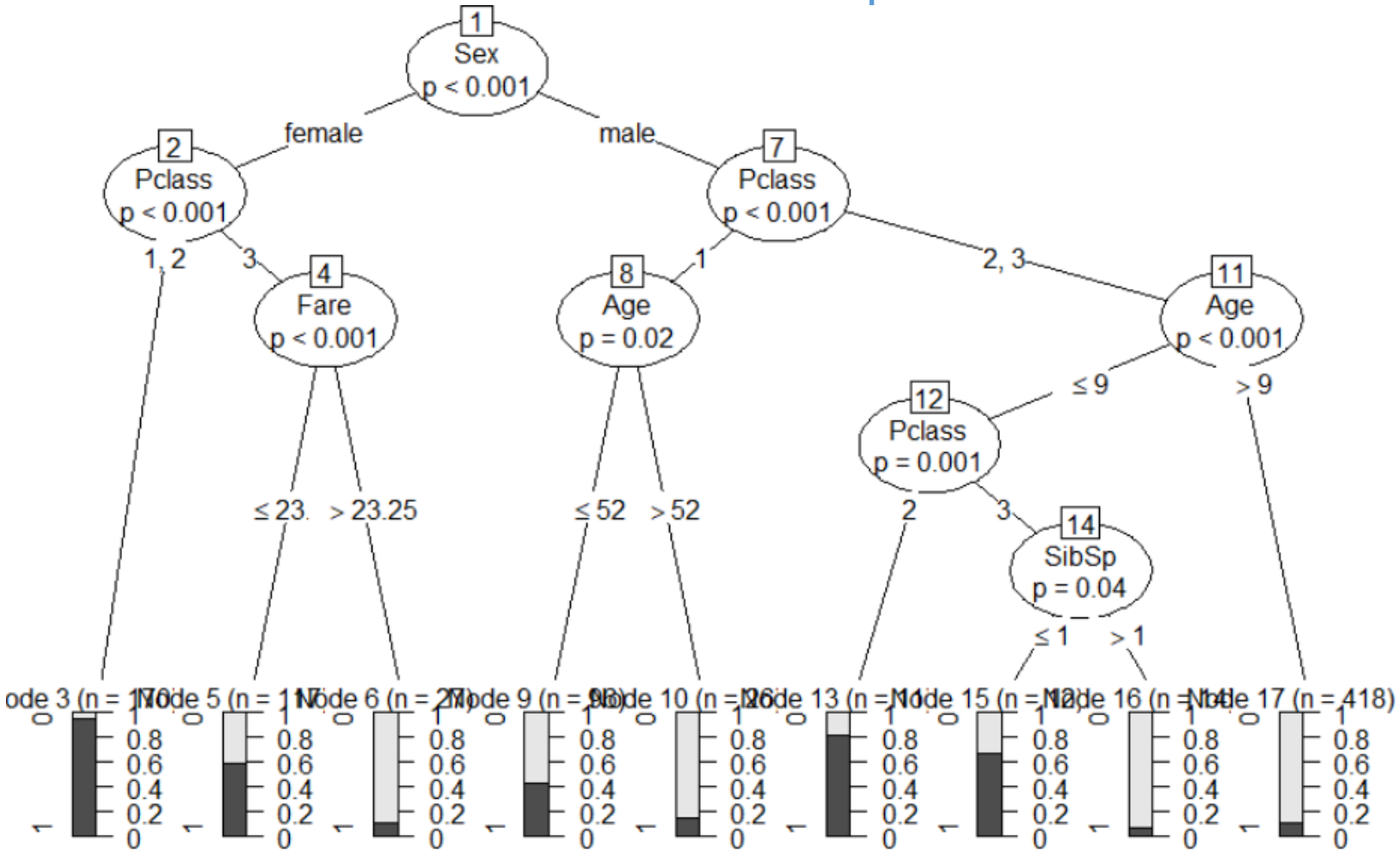
```
> titanic_tree

Model formula:
Survived ~ Sex + Pclass

Fitted party:
[1] root
|   [2] Sex in female
|   |   [3] Pclass in 1, 2: 1 (n = 170, err = 5.3%)
|   |   [4] Pclass in 3: 0 (n = 144, err = 50.0%)
|   [5] Sex in male
|   |   [6] Pclass in 1: 0 (n = 122, err = 36.9%)
|   |   [7] Pclass in 2, 3: 0 (n = 455, err = 14.1%)

Number of inner nodes:    3
Number of terminal nodes: 4
```

Estimate a tree model to predict titanic survival




```r
# estimate bigger model
titanic_tree_mod2 <- ctree(Survived ~ Sex + Pclass +
                                      Age + SibSp + Fare,
                           data = titanic_df)
titanic_tree_mod2

plot(titanic_tree_mod2)
```

Cross –Validate to Determine Optimal Tree Depth

```
# cross validate to get optimal tree depth
# must use rpart package here
library(rpart)
library(rpart.plot)

titanic_mod_rpart <- rpart(Survived ~ Sex + Pclass +
                             Age + SibSp + Fare +
                             Ticket + Cabin + Embarked +
                             Parch,
                           data = titanic_df,
                           method = "anova",
                           control = list(cp = 0,
                                          minsplit = 10,
                                          maxdepth = 5))


titanic_mod_rpart$cptable
plotcp(titanic_mod_rpart)
```