STATS 415: Tree-based methods

Prof. Liza Levina

Department of Statistics, University of Michigan

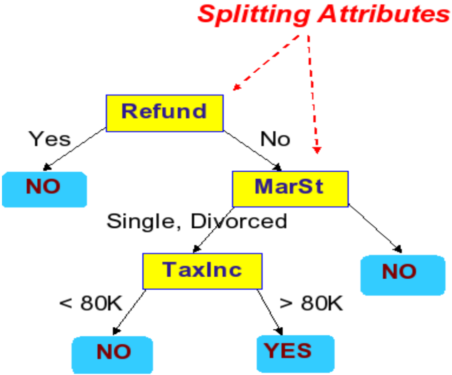
Tree-Based Methods

- Classification And Regression Trees (CART): build one tree
- Ensemble methods (bagging, random forests, boosting): combine many trees to improve performance

Example of a Classification Tree

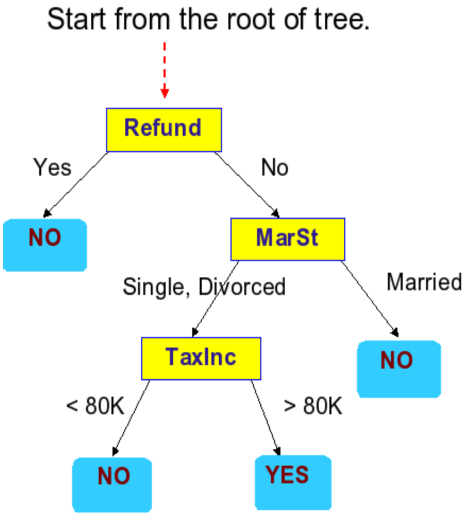
categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Model: Decision Tree

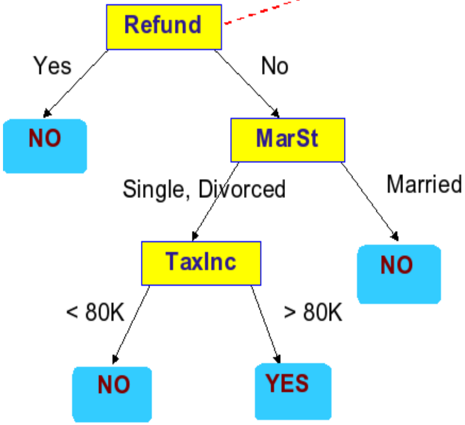
Classify Test Data



Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

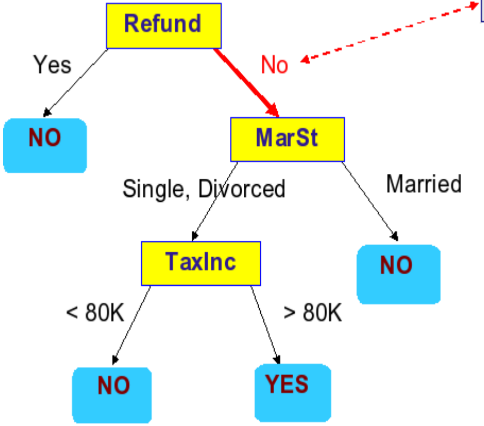
Classify Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

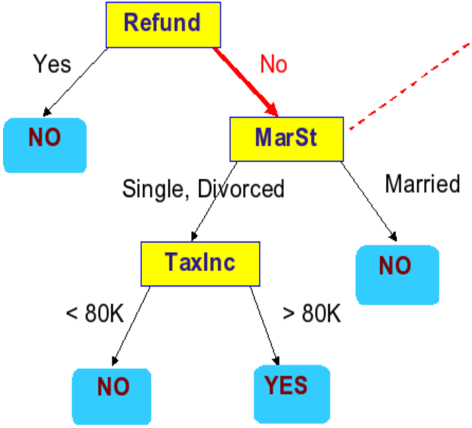


Classify Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



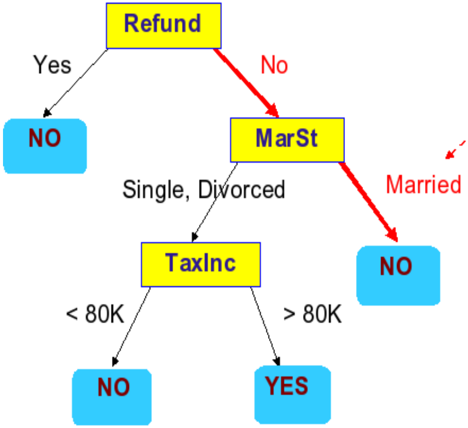
Classify Test Data



Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

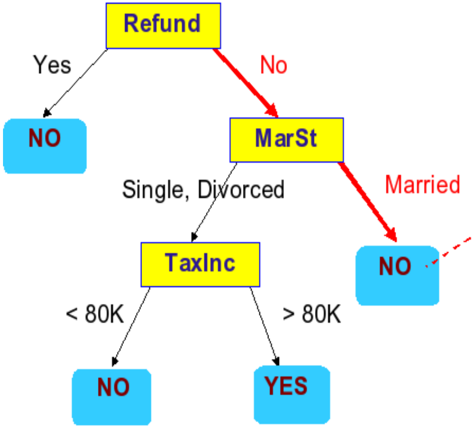
Classify Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Classify Test Data

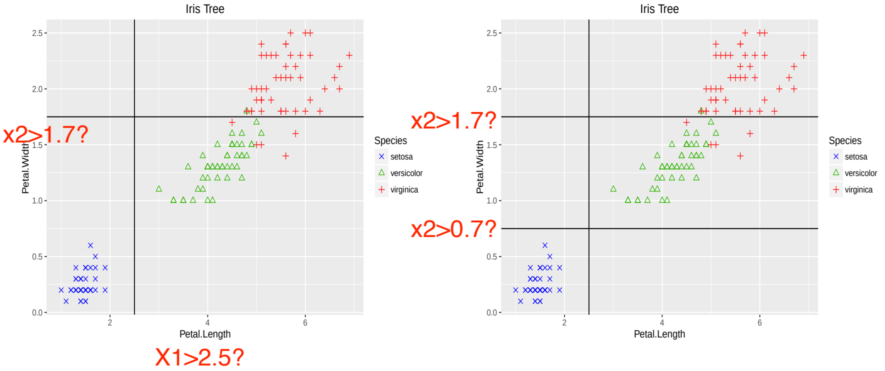
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Elements of a Tree

- Root node
- Splits #splits=#leaves-1
- Terminal node (leaf)
- Parent node
- Child node

We can have two trees work exactly the same.

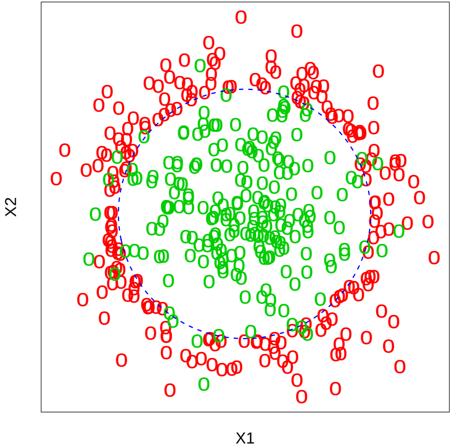


Another way to think about a tree: boundaries are determined by partitioning the range of one variable at a time

Classification Trees (CART)

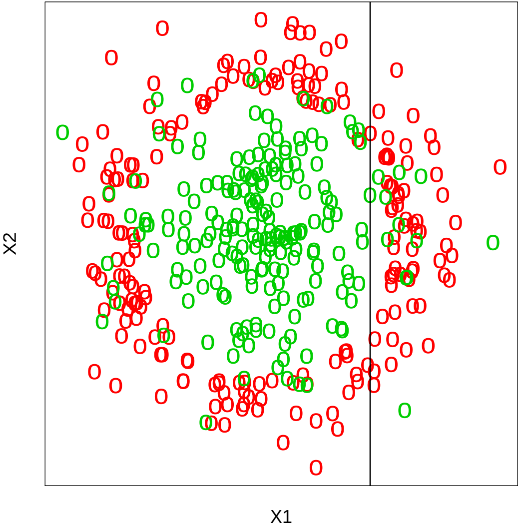
- Breiman, Friedman, Olshen & Stone (1984), Quinlan (1993)
- Recursively partition the input space into rectangular boxes
- Once the boxes are finalized, predict the class of each box by majority vote

Example: Nested Spheres

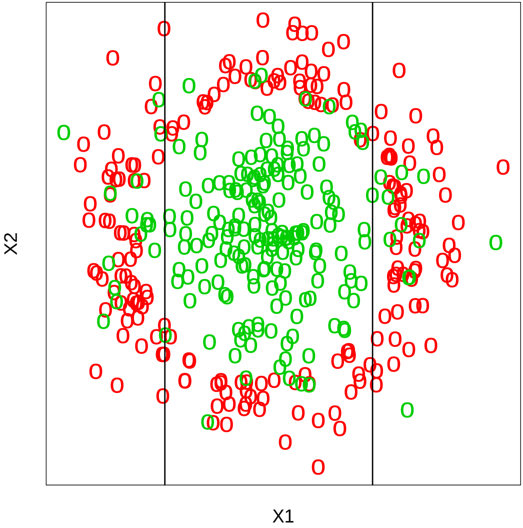


- Green class: two independent standard normal inputs X_1, X_2
- Red class: conditioned on $X_1^2 + X_2^2 \geq 4.6$

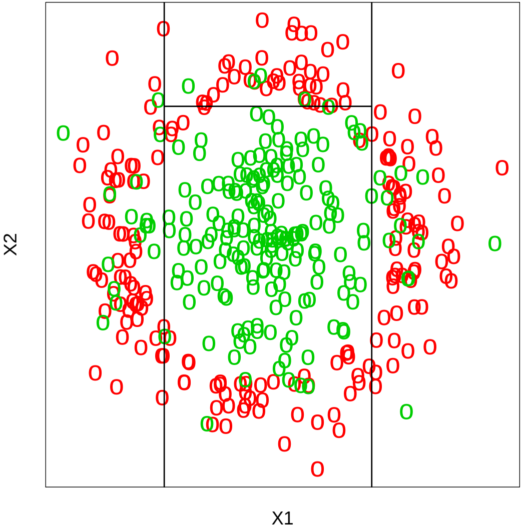
Classification Tree



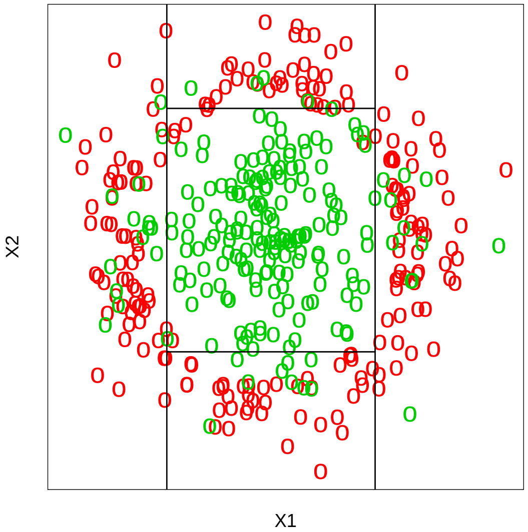
Classification Tree



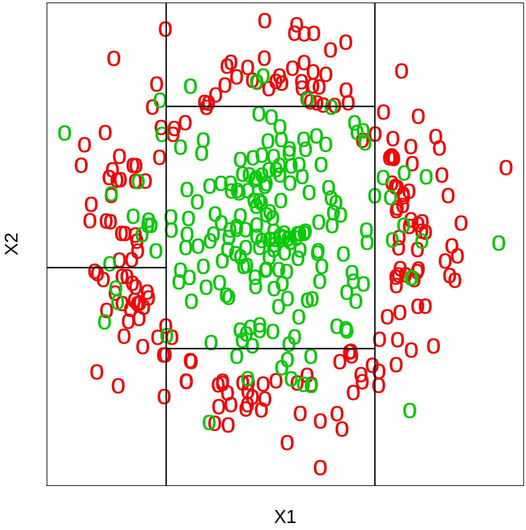
Classification Tree



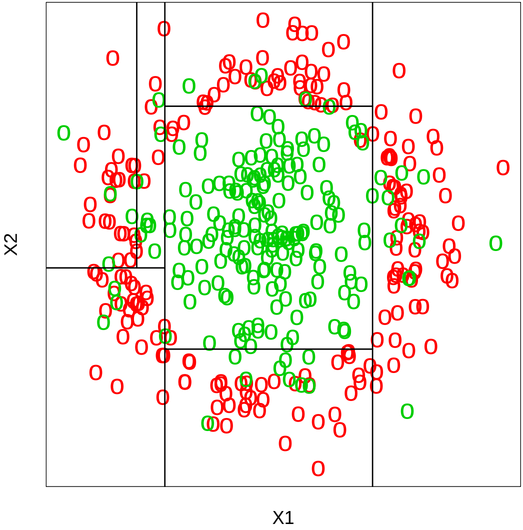
Classification Tree



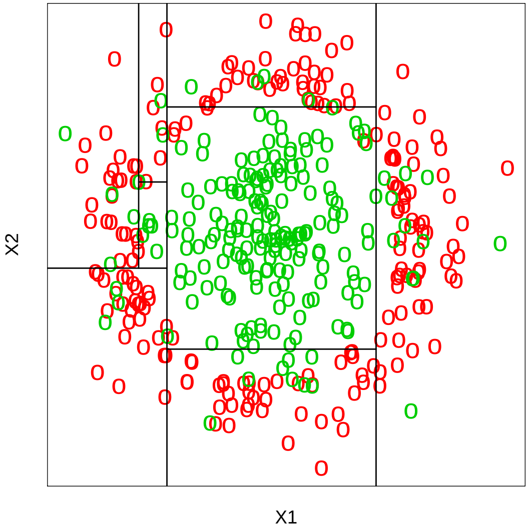
Classification Tree



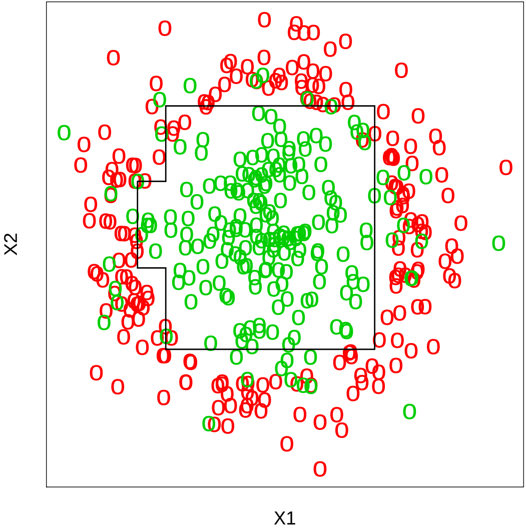
Classification Tree



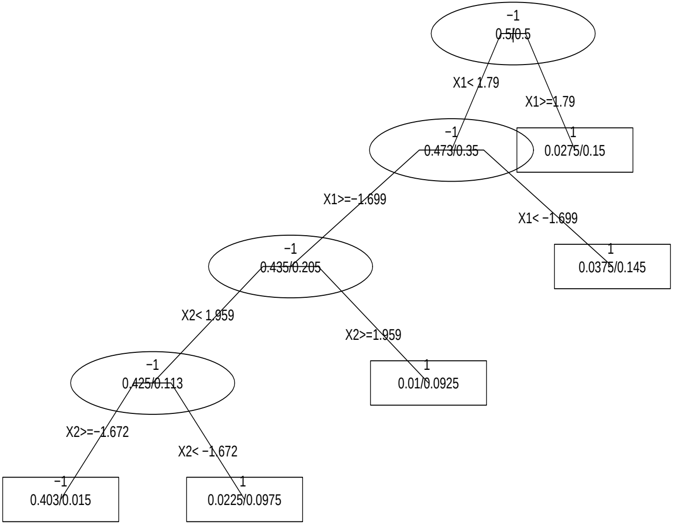
Classification Tree



Final Decision Boundary

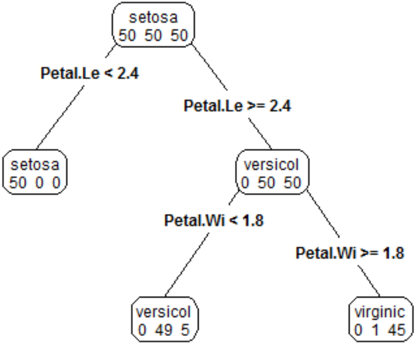


Classification Tree: Representation

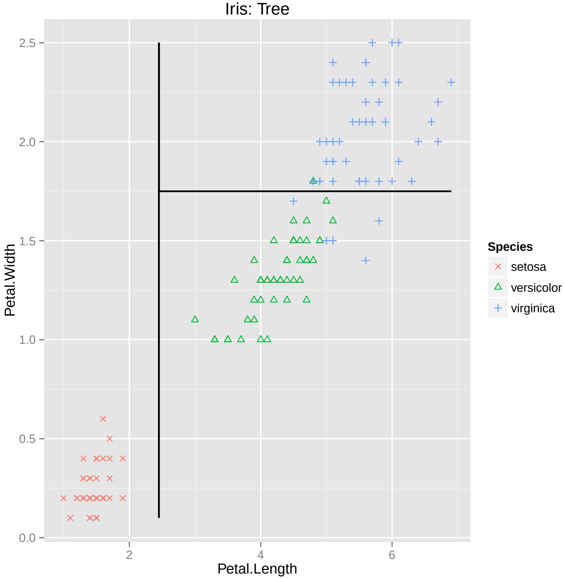


Example: Iris data set

```
> library(rpart)
> iristree = rpart(Species ~ ., data=iris)
> require('rpart.plot')
> prp(iristree,type = 4,extra = 1,clip.right.labs = F)
```



Tree boundaries



Main issues in building a classification tree

1. Splitting rules
2. Measuring split quality
3. Pruning rules
4. Overall optimality?
5. Stability (variance)?

Splitting rules

- Most implementations consider only binary splits – multiway splits tend to fragment the data too quickly.
- Let R_m denote the part of the feature space corresponding to node m, containing n_m observations. Define

$$p_k(m) = n_m^{-1} \sum_{x_i \in R_m} I(y_i = k),$$

  the proportion of observations from class k in region R_m.
- Observations in node m are classified to the majority class $k(m)$:

$$k(m) = \arg\max_k p_k(m)$$

Node impurity measures

How good is the node we just created? Need to quantify.

- Misclassification error

$$Q_m = n_m^{-1} \sum_{i \in R_m} \mathbf{1}(y_i \neq k(m)) = 1 - \max_k p_k(m)$$

- Gini index

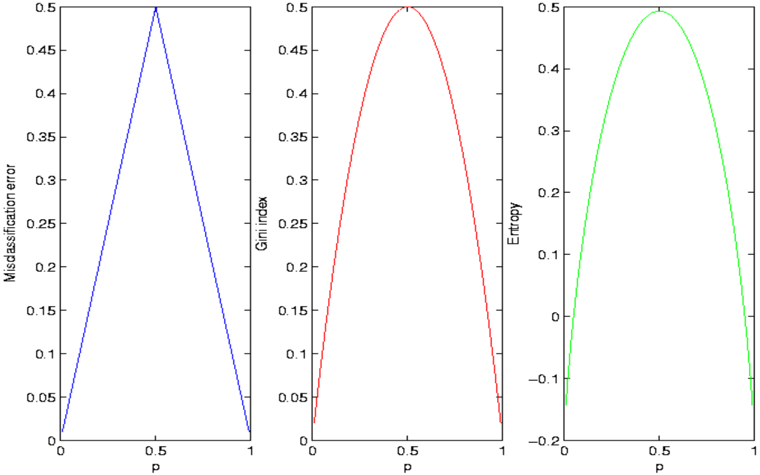
$$Q_m = \sum_{k=1}^{K} p_k(m)(1 - p_k(m)).$$

(the variance of a Bernoulli r.v. is $p(1-p)$)

- Entropy or deviance (another measure of variance)

$$Q_m = -\sum_{k=1}^{K} p_k(m) \log p_k(m).$$

Plots of impurity measures



Gini and Entropy are differentiable

Splitting Based on Impurity

- When considering which parent node P with n_P observations to split into two children nodes L and R (left and right) with n_L and n_R observations, respectively, we aim to maximize

$$M(\text{split}) = Q(P) - \left(\frac{n_L}{n_P} Q(L) + \frac{n_R}{n_P} Q(R) \right)$$

- This measures the average increase in “quality” (= decrease in impurity) from parent to children
- Greedy method – no global optimization

Misclassification Error vs Gini

		Cl1	Cl2	p_1	p_2	Gini	Error
	Parent	20	20				
Split 1	Left	10	20				
	Right	10	0				
Split 2	Left	15	5				
	Right	5	15				

Gini:
$M(\text{split 1}) =$
$M(\text{split 2}) =$
Error: $M(\text{split 1}) =$
$M(\text{split 2}) =$

Misclassification Error vs Gini

		Class1	Class2	p_1	p_2	Gini	Error
	Parent	20	20	0.5	0.5	0.5	0.5
Split 1	Left	10	20	1/3	2/3	4/9	1/3
	Right	10	0	1	0	0	0
Split 2	Left	15	5	0.75	0.25	0.375	0.25
	Right	5	15	0.25	0.75	0.375	0.25

Gini: $M(\text{split 1}) = 0.167$, $M(\text{split 2}) = 0.125$.
Classification error: $M(\text{split 1}) = M(\text{split 2}) = 0.25$

Gini tends to be more sensitive to node purity

Choosing tree size

- Tree size $|T|$ (number of leaves) is a measure of model complexity.
- Tree too large: overfitting
- Tree too small: may miss important structure and/or not classify well.
- Most algorithms grow a large tree, then prune it.

Cost-Complexity Pruning

For a given value of tuning parameter α ("complexity parameter") find a subtree T that minimizes the cost complexity criterion

$$C(T) = \sum_m n_m Q_m(T) + \alpha |T|$$

where the sum is over all terminal nodes, and Q is a node impurity measure

- $\alpha = 0$ corresponds to minimizing training error (will choose the largest tree)
- Large α results in small trees; small α results in large trees
- Choose α using cross-validation.
- $\alpha = 2(K-1)$ gives the Akaike Information Criterion (AIC)

Regression trees

- Same basic idea: divide the space into boxes by splitting on one variable at a time
- Within each box R_j, estimate the function by the average of all training point responses in that box, $\hat{y}_{R_j}$
- Looking to minimize, for final "boxes" $R_1, \ldots, R_{|T|}$

$$\sum_{j=1}^{|T|} \sum_{i: x_i \in R_j} (y_i - \hat{y}_{R_j})^2$$

- Minimizing on the training data without a complexity penalty will lead to overfitting (can split until each box has exactly one training point in it)

Fitting regression trees

- The cost-complexity criterion: minimize

$$\sum_{j=1}^{|T|} \sum_{i: x_i \in R_j} (y_i - \hat{y}_{R_j})^2 + \alpha |T|$$

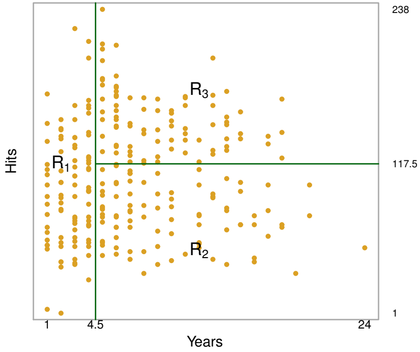
- Tuning parameter α can be selected by cross-validation
- Need a different notion of impurity:
  - Suppose we are considering splitting on a continuous variable X_j. Define

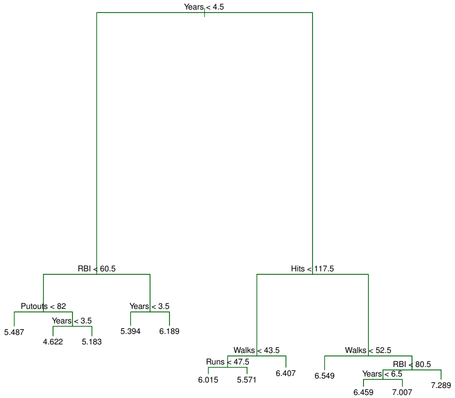
$$R_1(j,s) = \{X | X_j < s\} \ , \ \ R_2(j,s) = \{X | X_j \geq s\}$$

  - Then the split is obtained by minimizing, over all possible j, s

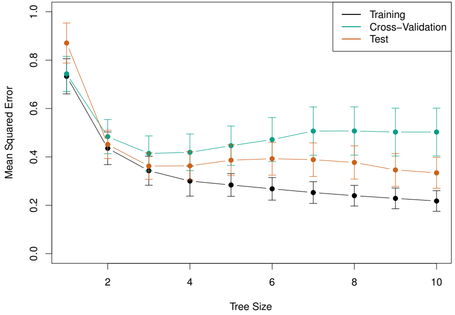
$$\sum_{i: x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1(j,s)})^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2(j,s)})^2$$

Example: predicting baseball players' salaries



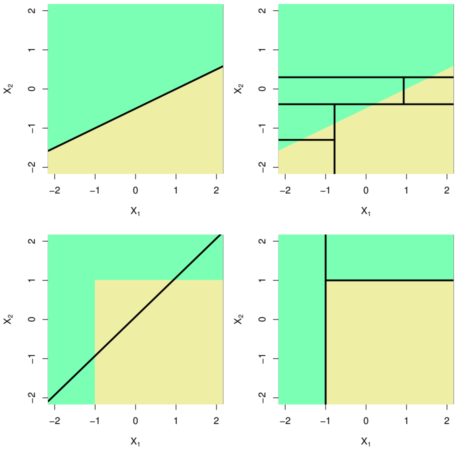


Years < 4.5
RBI < 60.5
Hits < 117.5
Putouts < 82
Years < 3.5
Years < 3.5
5.487
4.622
5.183
5.394
6.189
Walks < 43.5
Runs < 47.5
6.015
5.571
6.407
Walks < 52.5
6.549
RBI < 80.5
Years < 6.5
6.459
7.007
7.289



Training
Cross−Validation
Test
Mean Squared Error
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
Tree Size

Trees vs linear models



Which one is better will depend on the data

Strengths of trees

- Inexpensive to construct
- Variable selection
- Implicitly accounts for interactions
- Easy to interpret for small trees
- Can handle missing data
    - For categorical data, add a category "missing"
    - Can calculate "back-up" splits at each step that can be followed if a given predictor is missing
- Mixed variable types (categorical and quantitative)
- Invariant to monotone transformations of input variables

Weaknesses of trees

- Lack of global optimality (greedy)
- Instability: an error at the top of the tree is propagated through all levels, thus small changes in the data can lead to completely different trees
  - This difficulty can be alleviated using ensemble procedures such as bagging and boosting.
- Difficulty in modeling additive structures