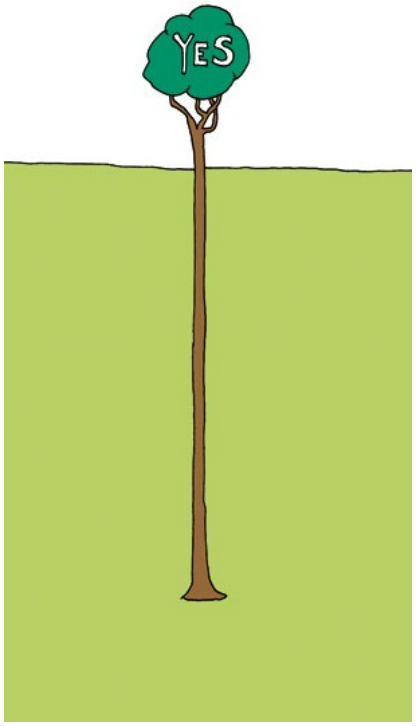
Decision Trees and Random Forests



Today

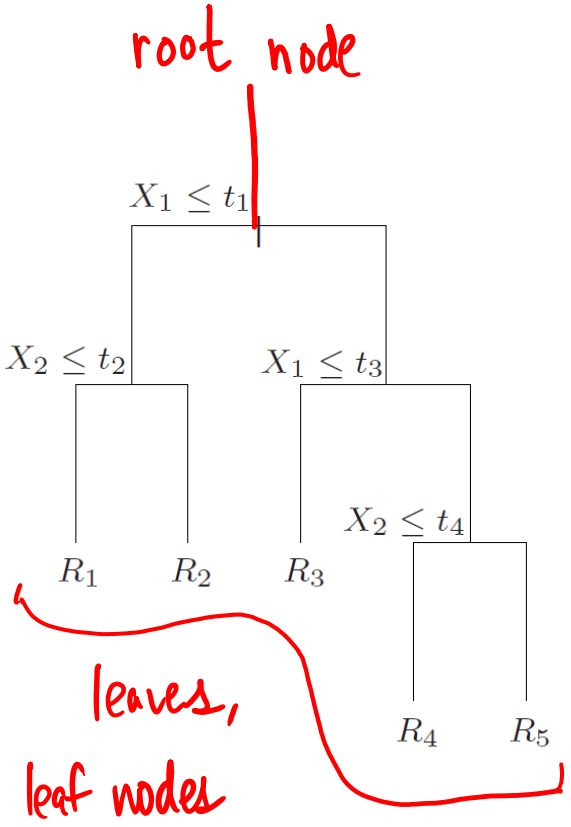
Decision trees

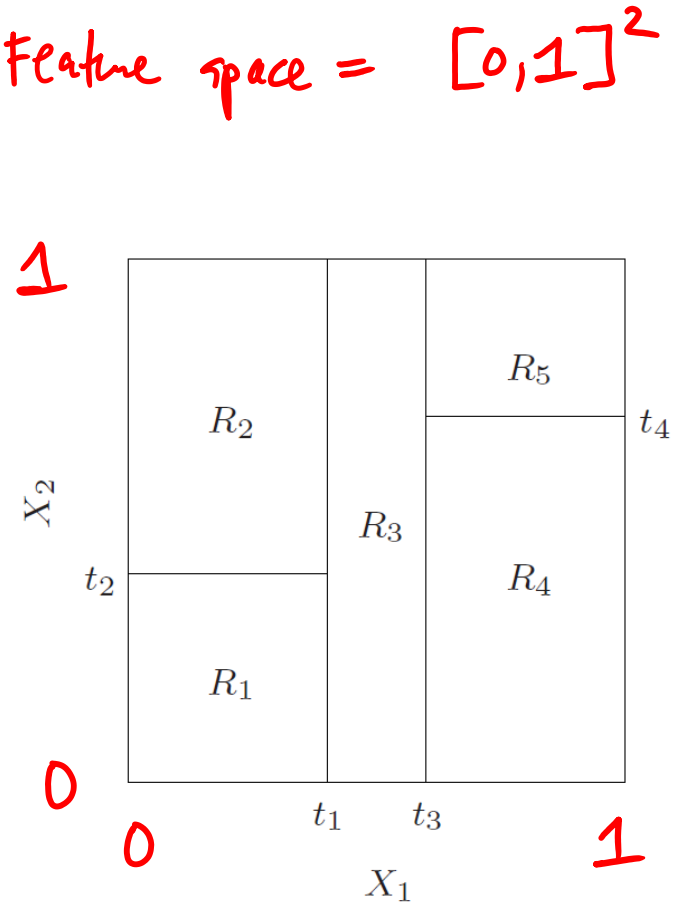
- Classification trees
- Regression trees

Random forests

The process for learning a decision tree from training data has connections with both information theory and empirical risk minimization

Tree-Structured Partitions of Feature Space



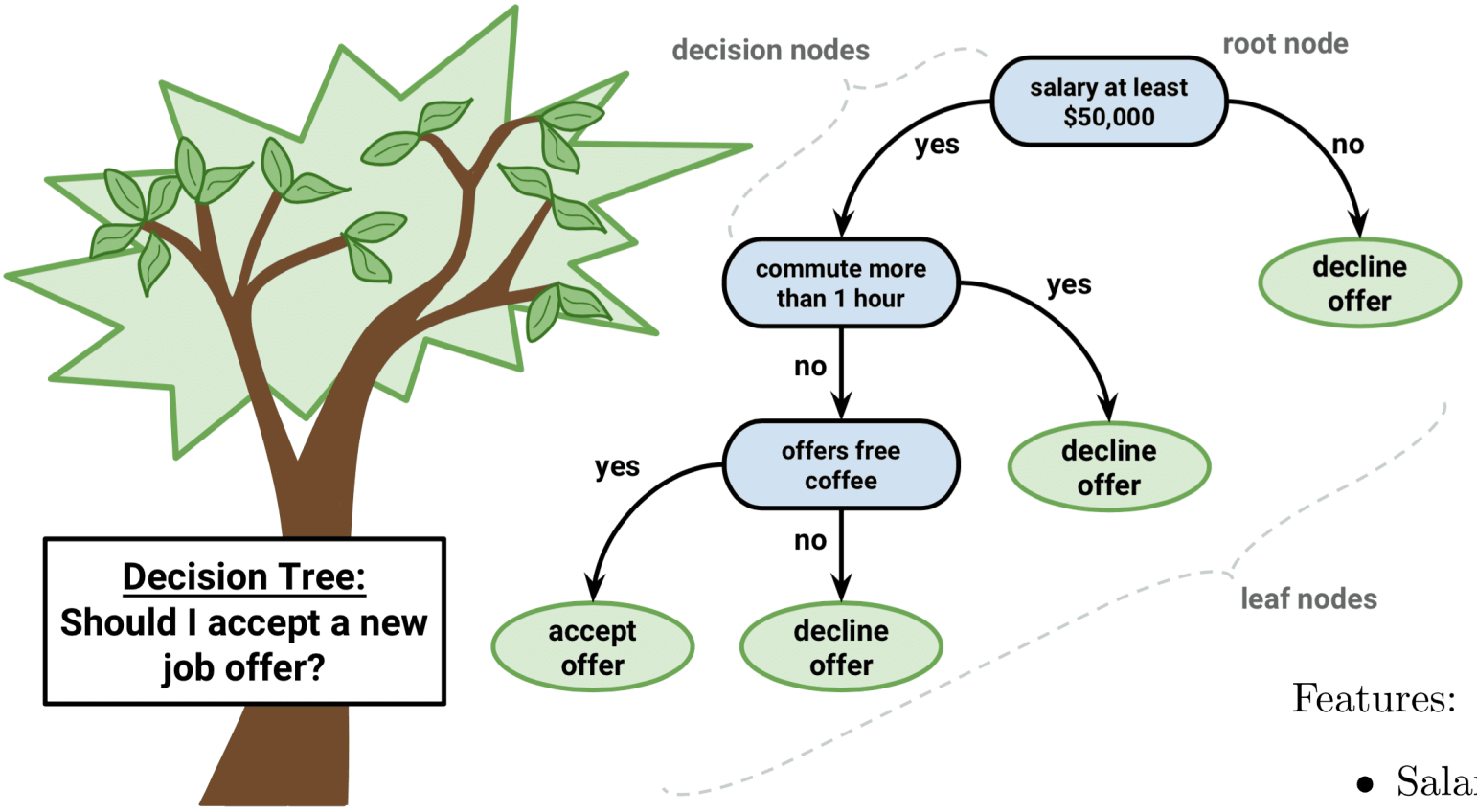


Figures from Hastie, Tibshirani, and Friedman

Decision Trees

- A **decision tree** is a tree-structured partition that is used for prediction
- There are two main types
- A **classification tree** is a tree-structured partition where each rectangle is associated with a class label
- A **regression tree** is a tree-structured partition where each rectangle is associated with a real number
- For both types of decision trees, prediction is accomplished by traversing the tree from the root to a leaf, and predicting the label/real number at that rectangle.

Example: Classification Trees

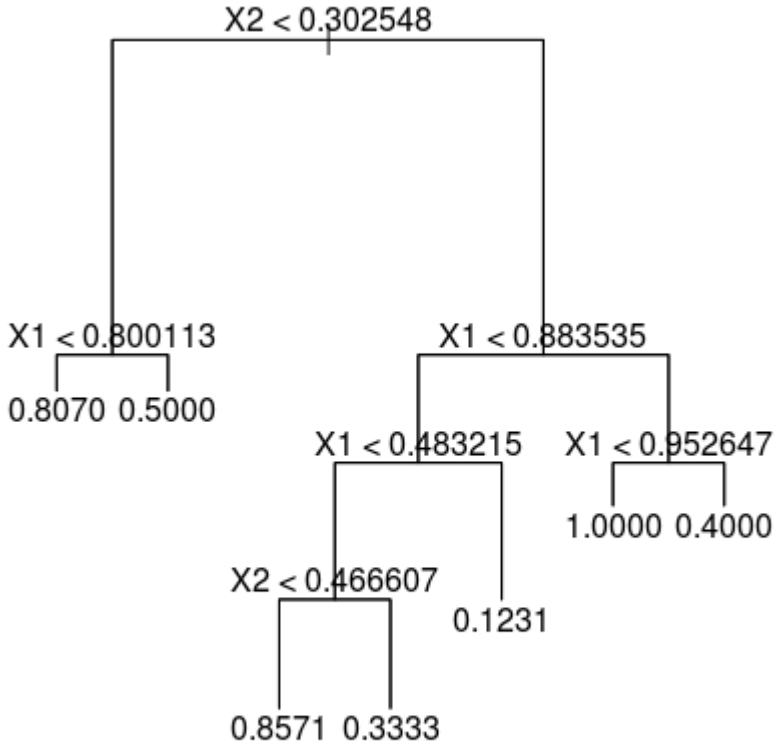


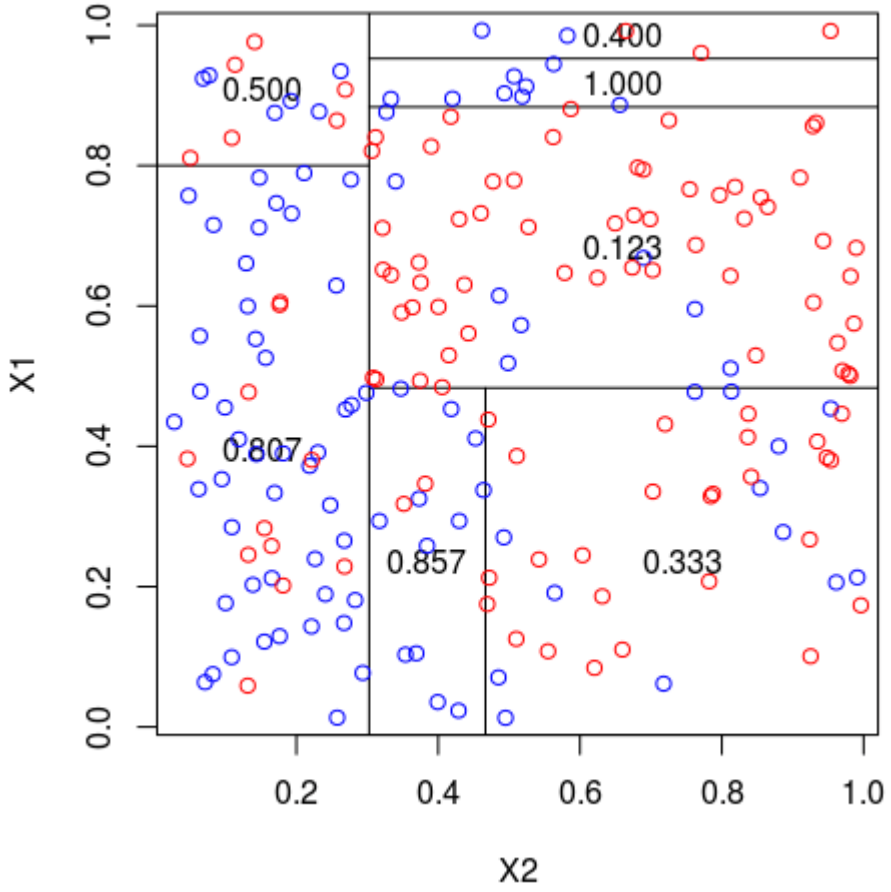
https://www.datacamp.com/tutorial/decision-trees-R

Features:

- Salary
- Commute time
- $\mathbf{1}_{\{\text{offers free coffee}\}}$
- Maybe others

Visualization of a Classification Tree



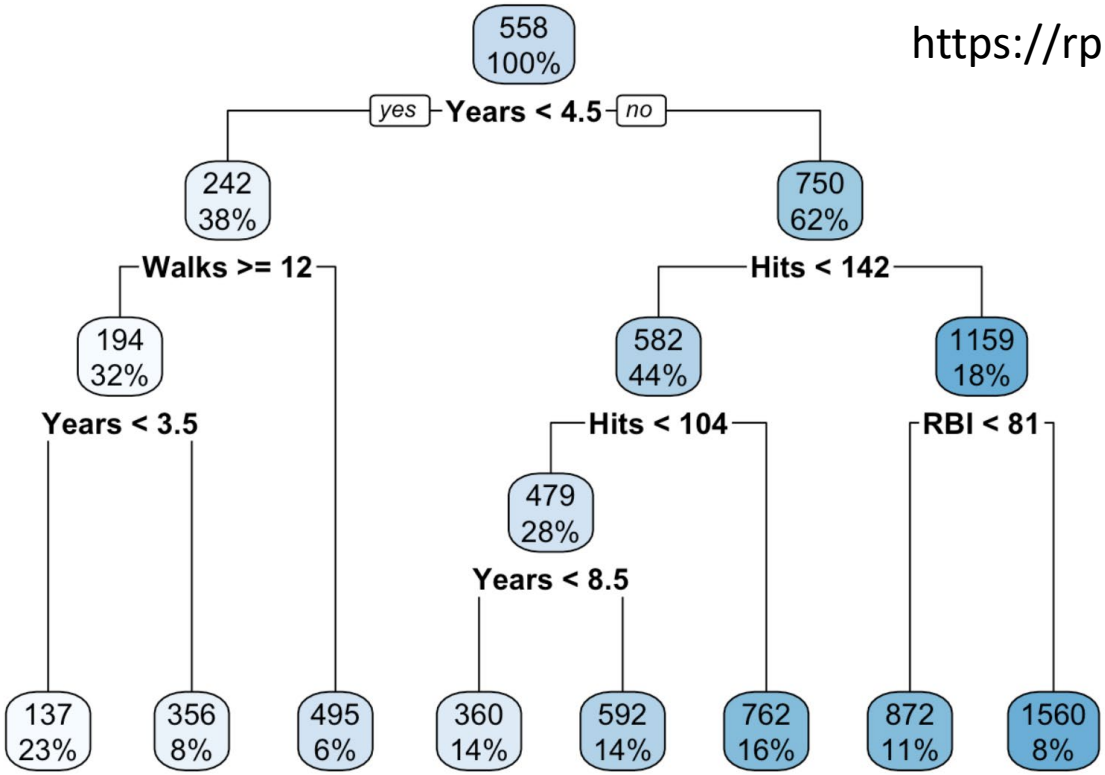


label = majority vote in each leaf

Number at leaf nodes = proportion of blue class in rectangle

https://www.datacamp.com/tutorial/decision-trees-R

Example: Regression Trees



https://rpubs.com/cyobero/regression-tree

Left = “Yes”
Right = “No”

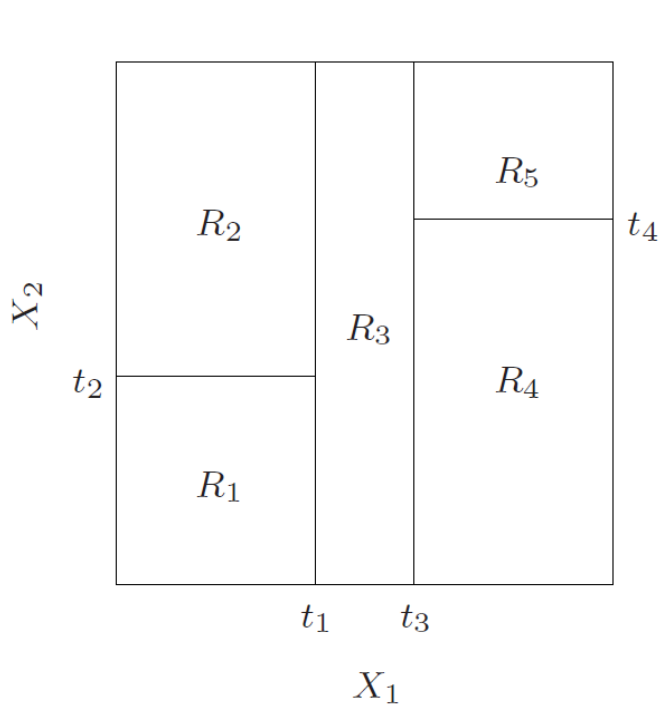
Features:

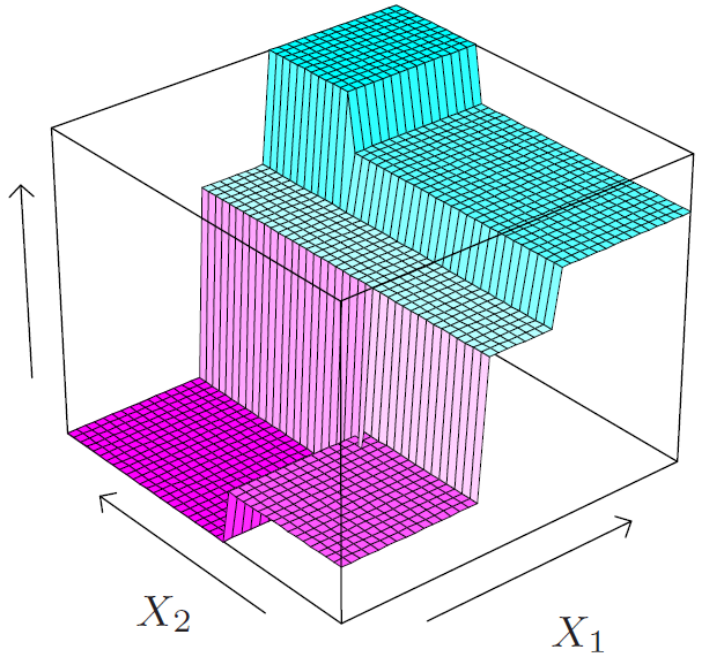
- Years = number of years in MLB
- Walks = number of walks drawn last season
- Hits = number of hits last season
- Maybe others

RBI >

Response variable:
Player salary

Visualization of a Regression Tree





Figures from Hastie, Tibshirani, and Friedman

Learning Decision Trees

- Given training data

$$(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$$

  for classification or regression, how should we construct a decision tree?

- Learning is broken into two phases: growing, pruning

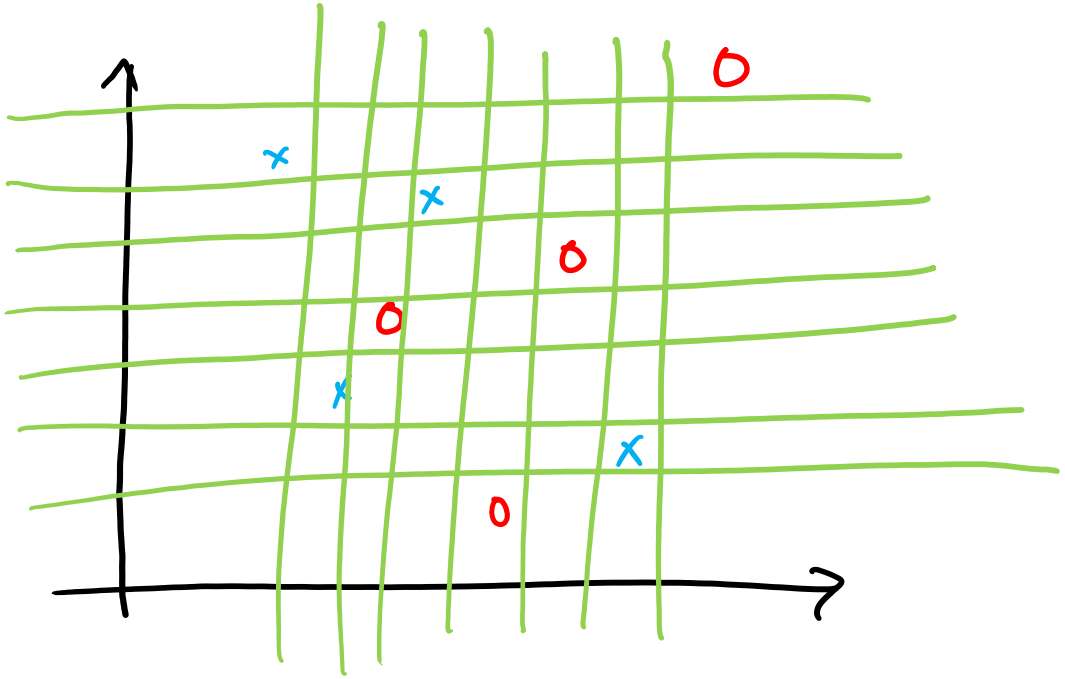
$$\min_{\{\text{all dec. trees } f\}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i))$$

NP-hard

Growing A Decision Tree

1. Identify candidate splits
2. Select best split

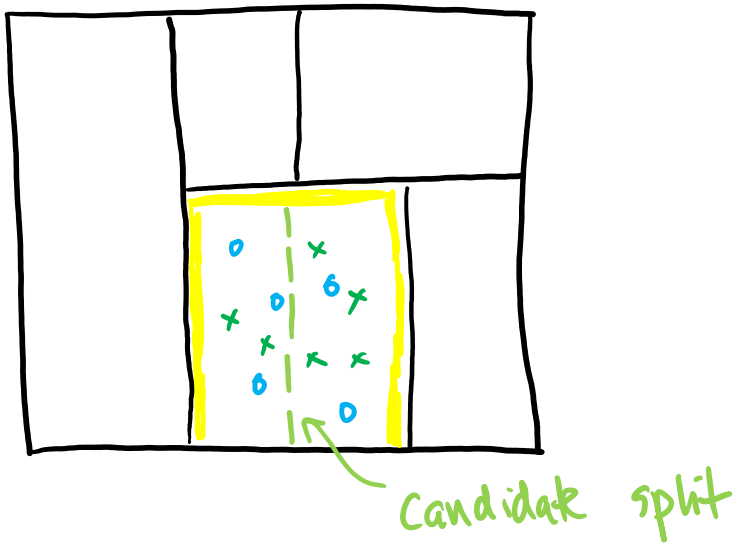
axis-orthogonal splits

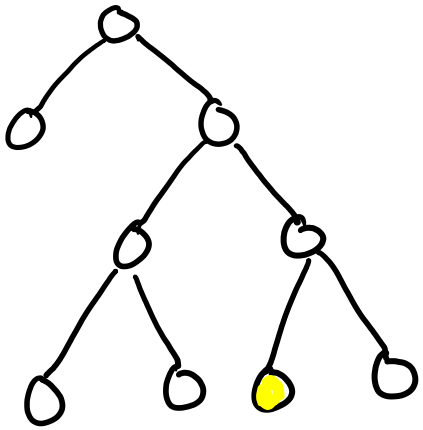


Split Selection: Binary Classification

- Suppose a candidate split partitions the data into two sets. Introduce the following notation

q_P	proportion of class 1 in parent node
q_L	proportion of class 1 in left child
q_R	proportion of class 1 in right child
p	proportion of data in parent node that goes to left child



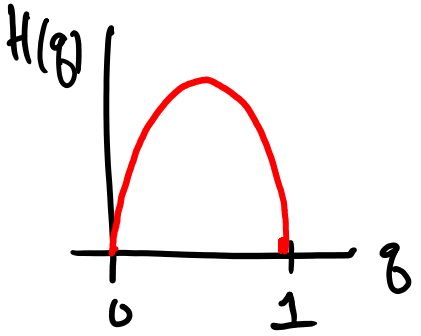


Split Selection: Binary Classification

- Define the *binary entropy function*

$$H(q) := -[q \log_2 q + (1-q) \log_2 (1-q)]$$

function



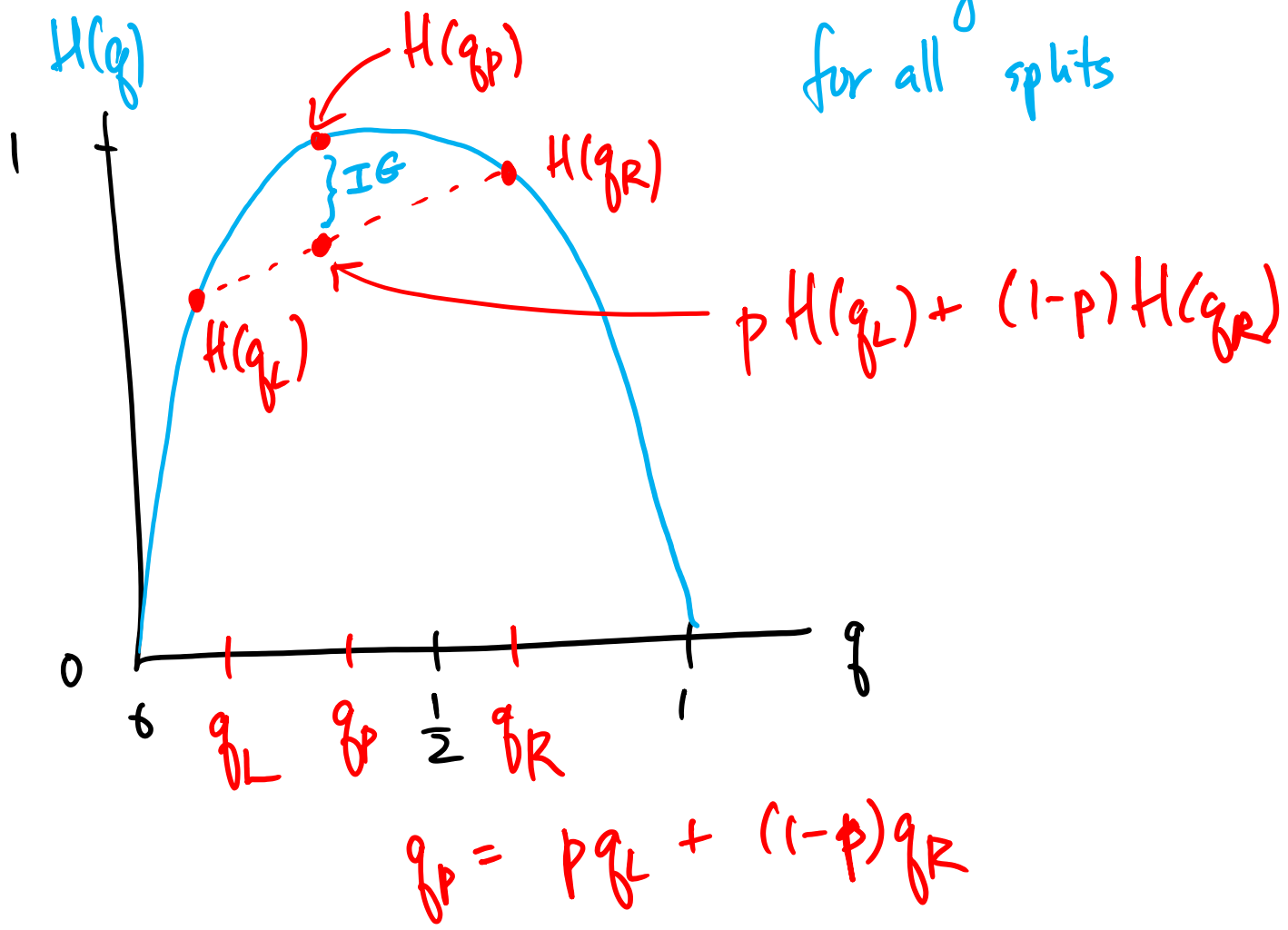
- The *information gain* of a split is defined as

$$IG = H(q_P) - [p\, H(q_L) + (1-p)\, H(q_R)]$$

- $H(q)$ may be viewed as a measure of node impurity
- Select the split with the largest information gain
- Repeat the process on each children
- Each node is given the label of the majority class in that rectangle

Information Gain

concavity: $IG \geq 0$ for all splits



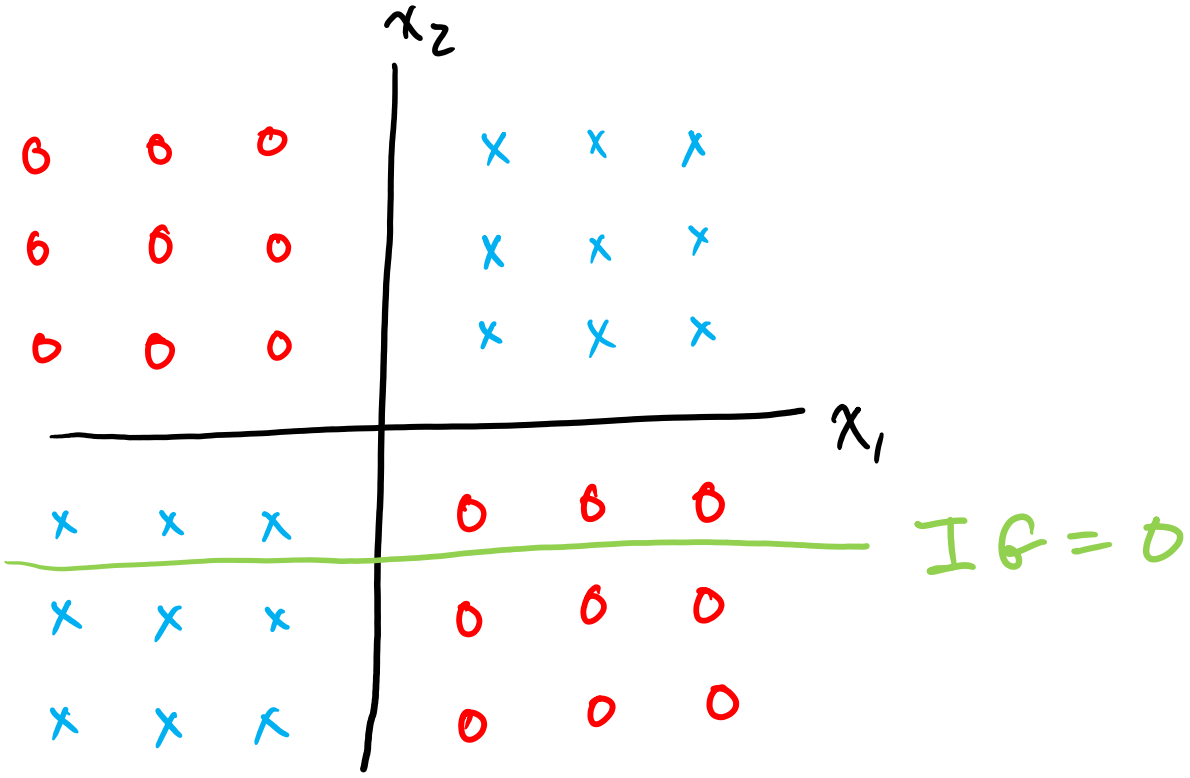
$q_P = p q_L + (1-p) q_R$

Poll

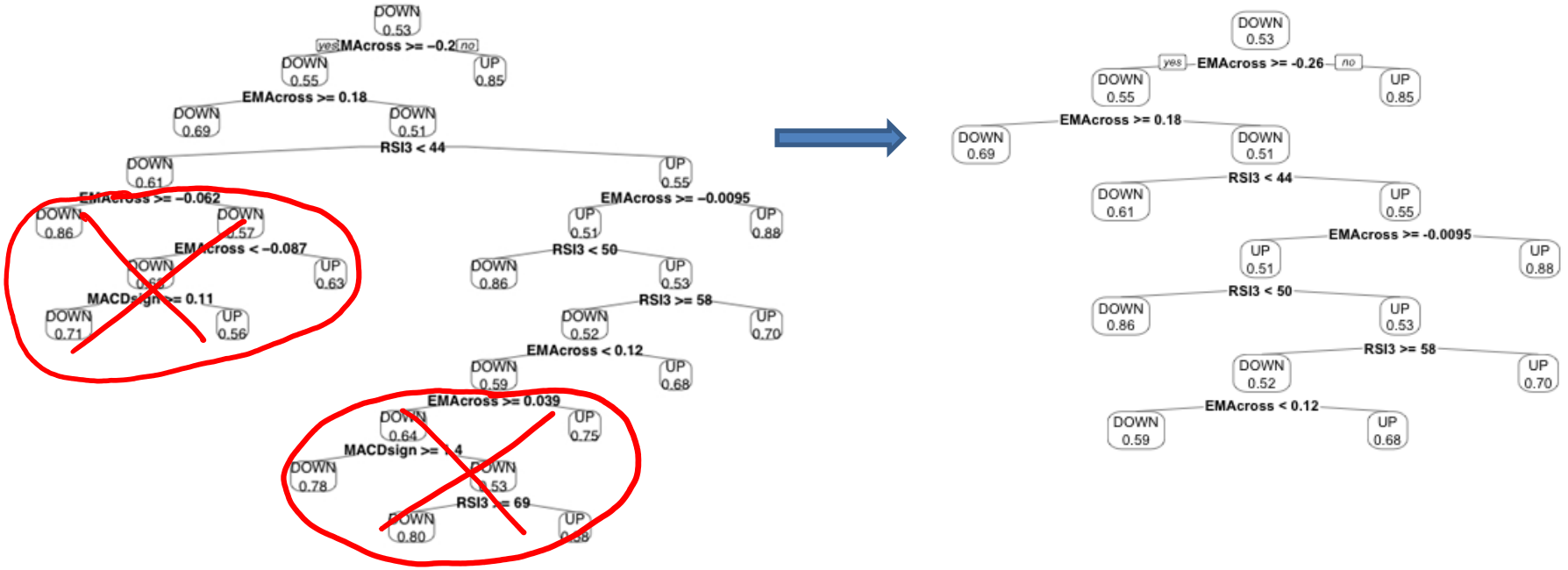
True or false: It is a good idea to stop growing a classification tree when the best information gain of any split is below some small, positive threshold

(A) True
(B) False

Ancillary Splits



Pruning



Typically one grows a large tree that perfectly fits the training data, and then **prunes** it back to avoid overfitting.

Cost-Complexity Pruning

- $T_0 =$ initial tree (after greedy growing)
- Pruned subtree

$$\widehat{T} = \underset{T \subseteq T_0}{\arg\min} \; \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{\{T(\boldsymbol{x}_i) \neq y_i\}} + \lambda |T|$$

  where

  - $T \subseteq T_0$ indicates search over all *pruned subtrees* of T_0
  - $|T|$ is the number of leaf nodes
  - $\lambda > 0$ is a regularization parameter

- This criterion can be optimized efficiently (no surrogate loss needed)

subtrees containing root

Extensions

Multiclass Classification:

$q = (q_1, \ldots, q_c)$

Just replace binary entropy with entropy

$$H(q) = -\sum_{i=1}^{c} q_i \log q_i$$

Regression:

Label each node with average response value among data reaching that node

Growing: replace IG by decrease in average squared prediction error.

Pruning: cost-complexity pruning with squared error loss

Decision Trees: Summary

Strengths:

- Interpretable, expressive
- Fast (learning + prediction)
- Categorical features

Weaknesses:

- Training is suboptimal
- Sensitivity to perturbations of training data

Ensemble Methods

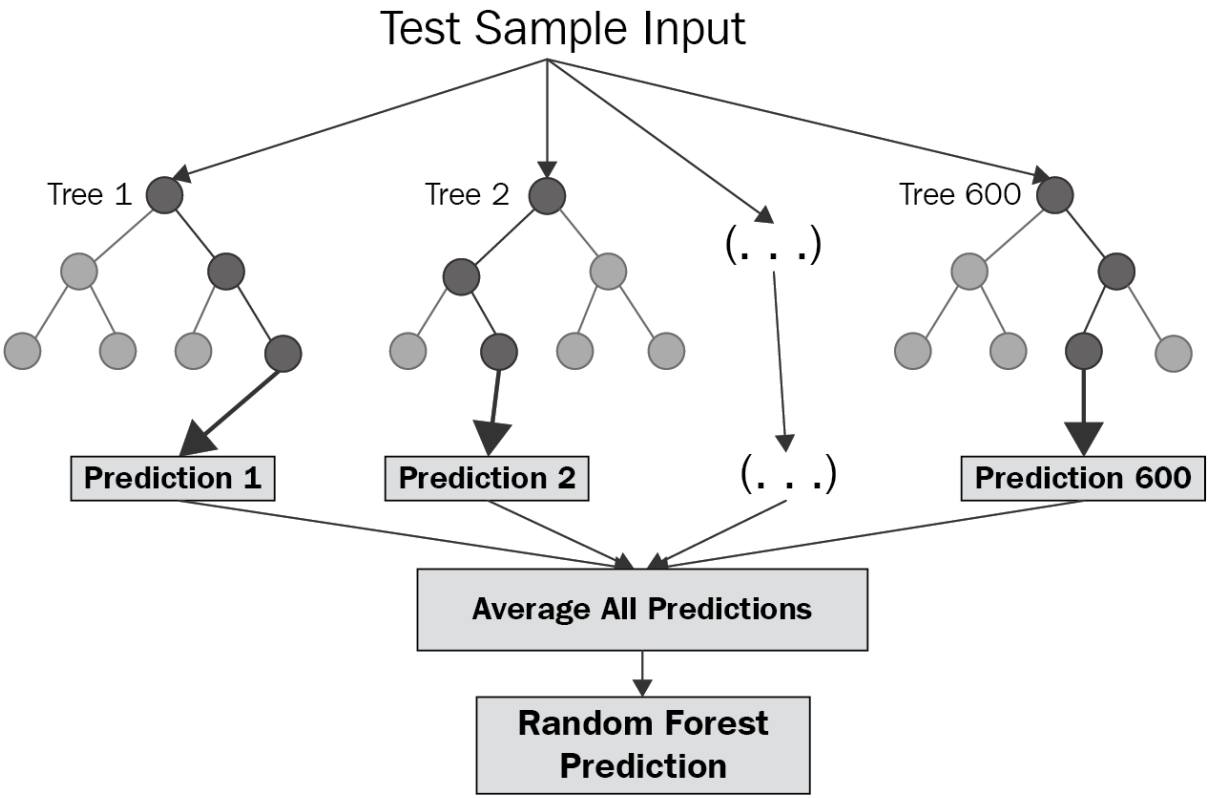
Random forests and boosting are examples of **ensemble methods**: They obtain a prediction function by taking several different prediction functions (classifiers or regression estimates) and averaging* the results.

Ensemble methods are usually coupled with **decision trees**. Why trees?

- Trees are very expressive: Deep trees represent complex decision boundaries
- Trees have high variance (sensitivity): Randomization can lead to very different trees
- Averaging leads to variance reduction, meaning trees benefit a lot from averaging
- Trees can be learned very efficiently and can handle both numerical and categorical features at the same time

* mode of predicted labels in classification

Random Forests



Random Forests

The basic idea behind random forests is straightforward:

Given the training data, generate *L* different decision trees by introducing randomness into the learning process

Examples of randomness

- Base your decision tree on a random subsample of the full sample
- Select a random subset of features on which to construct your tree
- Randomly transform data
- …

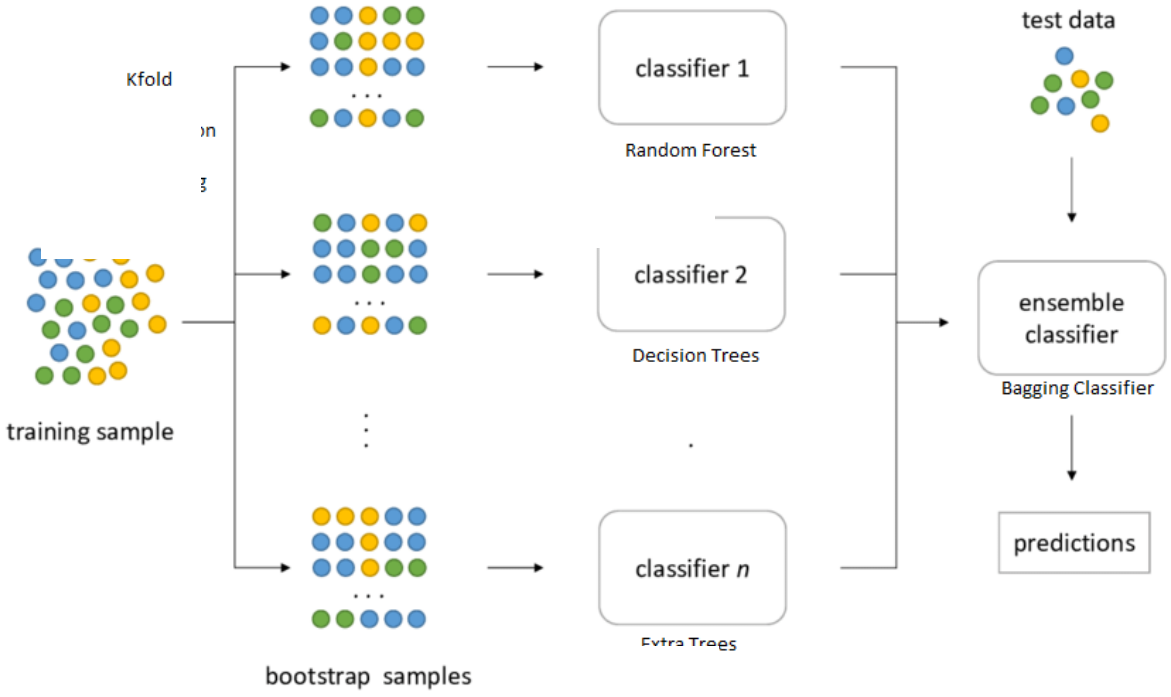
Having generated *L* different decision trees, the random forest predictor (classifier or regression estimate) is the average of these randomly generated decision trees.

Random forests achieve state-of-the-art prediction for many datasets. They overcome the sensitivity issue, at the expense of interpretability.

Random Forests

Base each tree on a bootstrap sample:

- Recall a boostrap sample is a random sample obtained by sampling with replacement, with the same size as the original training data
- If this is the only source of randomness, it is called “Bagging” for “bootstrap aggregation”



Bagging Classifier Process Flow

Random Forests

Base each split on a random subset of features:

- Avoids having all trees start out the same way (in the case of one or a few very discriminating features)
- Decreases correlation between trees
- Hence, final classifier has less variance, being the average of de-correlated trees
- Tuning parameters:
  - Feature subset size
  - Max tree depth
  - Tree pruning penalty (although RFs also work well with no pruning)

Both bootstrap samples and random feature subsets are used in standard implementations of random forests

# of trees in ensemble

Leo Breiman

Breiman, Friedman, Olshen and Stone, *Classification and Regression Trees*, 1984

Breiman, “Random forests,” *Machine Learning,* vol. 45, pp 5-32, 2001

Breiman, “Bagging predictors,” *Machine Learning,* vol. 24, pp 123-140, 1996

Breiman, "Statistical Modeling: the Two Cultures,“ *Statistical Science*, 2001

Summary

- Decision trees: Interpretable prediction method based on recursive partitioning of feature space
- Can handle numerical and categorical attributes seamlessly
- Greedy growing
  - Classification: information gain
  - Regression: decrease in average squared error
- Cost-complexity pruning: A form of empirical risks minimization
- Random forests: average of randomly constructed decision trees
- To learn more: Hastie, Tibshirani, and Friedman
- Fun fact: Random forests used in Xbox 360 (Kinect sensor)