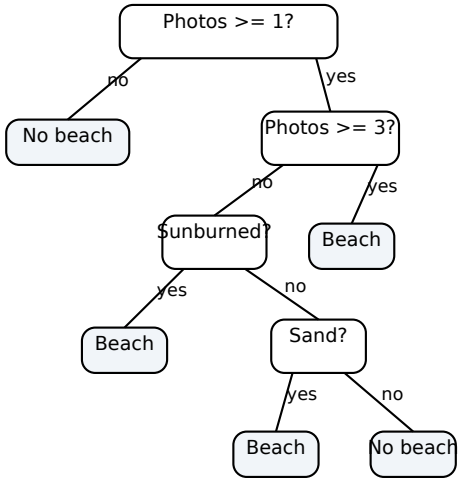
Decision trees

Spring break detective: did they go to a beach?

Small sample dataset and the fully developed tree.

Photos	Sun?	Sand?	Beach?
0	no	no	no
0	yes	no	no
1	no	no	no
1	no	yes	yes
2	yes	no	yes
2	no	yes	yes
3	no	no	yes
4	yes	yes	yes



Decision tree: what we want

- Goal 1: Compact - use as few nodes as possible while still separating classes.
- Goal 2: Pure leaves - leaf impurity near 0 means predictions are more reliable.
- Finding the smallest tree that classifies best is NP-hard.
- Practical fix: at each step choose the feature and threshold that most reduces impurity.

Consistency question

A consistent tree exists exactly when there are no duplicate feature vectors carrying conflicting labels. If identical inputs have different labels, no deterministic tree can classify them both correctly.

Impurity function of a dataset

Impurity of a split

```
S = {(x_1,y_1), ..., (x_n,y_n)},   y_i in {1, ..., c}
I(S) measures how mixed the labels are in S.
I_T(S) = (|S_L| / |S|) I(S_L) + (|S_R| / |S|) I(S_R)
S_L = {(x,y) in S : x_f <= t},   S_R = {(x,y) in S : x_f > t}
```

- Each part of a split can have a different impurity.
- Choose the split that minimizes the weighted average impurity of the left and right children.
- Tree learning is driven by local impurity reduction.

How do we measure "purity"?

For a subset S, let p_k be the fraction of examples in class k.

Max impurity

```
I_M(S) = 1 - max_k p_k
```

Error of Bayes optimal.

Gini impurity

```
I_G(S) = sum_k p_k (1 - p_k)
```

Binary case: impurity is largest near a 50/50 mix and drops to 0 at a pure leaf.

Entropy

```
I_H(S) = - sum_k p_k log p_k
```

Entropy penalizes uncertainty. Pure leaves have zero entropy; balanced leaves have larger entropy.

Variance

```
I_V(S) = Var(y_1, ..., y_|S|)
```

Use variance (or squared loss) for regression trees.

ID3 algorithm

```
ID3(S):

  If all examples in S have the same label:
      return a leaf with that label

  If no valid split is available:
      return a leaf with the majority label in S

  For each candidate split (f, t):
      S_L = {(x, y) in S : x_f <= t}
      S_R = {(x, y) in S : x_f > t}
      Compute I_T(S; f, t)

  Choose (f*, t*) with smallest I_T(S; f*, t*)
  Create node with test x_f* <= t* ?
  Left child  = ID3(S_L)
  Right child = ID3(S_R)
  return the resulting tree
```

Why not stop when no immediate impurity improvement happens? Some useful structures only emerge after multiple coordinated splits. The XOR pattern is the canonical example: no single first split solves it, but a depth-2 tree does.

CART: the same structure, but with continuous targets

For regression, leaves predict numeric values. The impurity is squared loss, and the leaf prediction is the mean target value in that region.

Regression-tree impurity

```
L(S) = (1 / |S|) sum_(x,y in S) (y
```

A split is good if it creates children whose values are tightly concentrated around their own means.

Prediction at a leaf

```
y_hat(x) = ybar_leaf(x)
```

The learned function is piecewise constant: each leaf corresponds to a region with its own average response.

Classification view

Leaf stores a class label or class histogram.
Output = majority label

Regression view

Leaf stores a real number, usually the mean tar
Output = local mean

Overfitting with decision trees

- If we allow the tree to grow to large depth, it can fit the training data extremely well but also overfit.
- Mitigation 1: constrain the maximum depth.
- Mitigation 2: constrain the minimum leaf size.
- Next lecture: allow deep trees but ensemble many of them (for example, random forests / boosting).