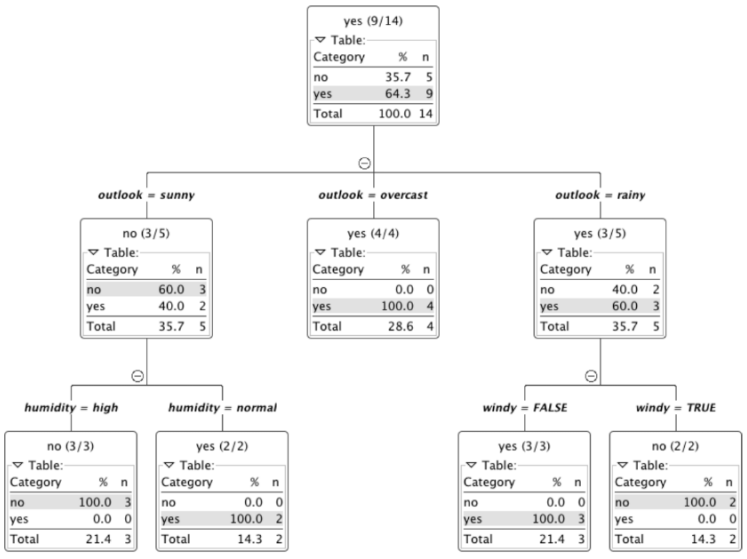
✱ Decision Trees



Decision Tree Representations (Graphical)

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

The Weather Dataset

- **Branch**: represents an outcome of the test (outlook=windy)
- **Node**: represents a class label or class label distr.
- **At each node**; one attribute is chosen to separate training examples of different classes

- **Top-down approach**: Initially, all training examples are at the root. Then, examples are recursively partitioned by choosing one attribute at a time

- **Bottom-up approach**: Remove subtrees or branches in a bottom-up manner to improve the estimated accuracy on new cases

» Which Attribute for Splitting?

- At each node, available attributes are evaluated based on separating the classes of training examples using either purity or impurity measures

- Typical measures: • Information Gain • Information Gain Ratio • Gini Index

1) Information Gain (ID)

- It increases with the **average purity** of subsets that an attribute produces. (Selects attribute with **highest gain**)

$$entropy(p_1, \ldots, p_n) = -p_1 \log_2 p_1 - p_2 \log_2 p_2 \ldots - p_n \log_2 p_n$$
or $info(D)$

→ entropy refers to amount of info expressed in bits

- ID is difference between the info before split and info after split

$$gain(A) = info(D) - info_A(D)$$

$$info_A(D) = \frac{|D_1|}{|D|} info(D_1) + \ldots + \frac{|D_n|}{|D|} \cdot info(D_n)$$

D_n: how many (yes+no) for n^{th} value of that attribute

D: how many (yes+no) in total

Ex

- "outlook" = "sunny" $info([2,3]) = entropy(2/5, 3/5) = 0.971$ ($-\frac{2}{5}\log_2(\frac{2}{5}) - \frac{3}{5}\log_2(\frac{3}{5})$)
- "outlook" = "overcast" $info([4,0]) = entropy(1,0) = 0.000$ ($-\log_2 1^0 - 0 = 0$)
- "outlook" = "rainy" $info([3,2]) = entropy(3/5, 2/5) = 0.971$
- Expected information for attribute $info([2,3][4,0][3,2]) = \frac{5}{14} \times 0.971 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.971$

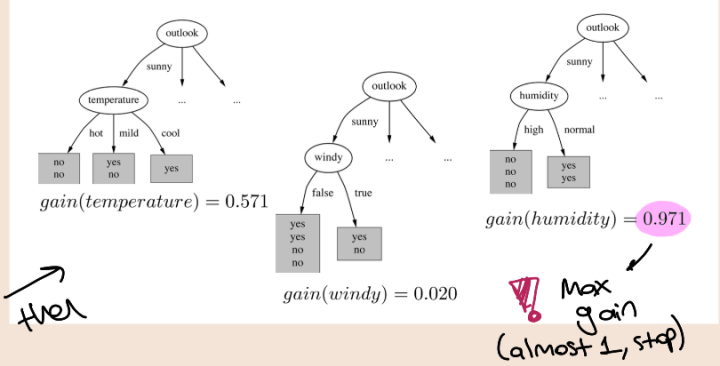
$gain(outlook) = info([9,5]) - info([2,3][4,0][3,2])$

$= 0.940 - 0.693 = 0.247$ → max gain (first split)

$gain(temperature) = 0.029$

$gain(humidity) = 0.152$

$gain(windy) = 0.048$



then

Max gain (almost 1, stop)

- When should building stop?
  - if all samples for given node belong to same class
  - there are no remaining attributes
  - there are no samples left
  - if there is nothing to gain

- Attributes with large # of values are problematic. IG rewards many-valued attributes
  Ex; id, primary-key ...
  ⇓
  Overfitting

2) Information Gain Ratio (IGR)

- Modification of IG that reduces the bias toward highly branching attributes
- It should be large when data evenly spread, small when all data belong to one branch
- It corrects IG by taking the intrinsic (özgün) information of a split into account

$$\text{IntrinsicInfo}(S,A) = -\sum \frac{|S_i|}{|S|} \cdot \log \frac{|S_i|}{|S|} \quad \rightarrow \quad \text{GainRatio}(S,A) = \frac{\text{Gain}(S,A)}{\text{IntrinsicInfo}(S,A)}$$

Ex:

Outlook	
Information after split:	0.693
Gain: 0.940-0.693	0.247
Split info: info([5,4,5])	1.577
Gain ratio: 0.247/1.577	0.156

bu ikisini bulmuştuk

number of; sunny, overcast, rainy → 5, 4, 5

- ID has even greater IGR → standard fix is an ad-hoc test to prevent splitting
- In practice, tree learners use both IG and IGR

3) Gini Index

$$gini(T) = 1 - \sum_{j=1}^{n} p_j^2$$

T : dataset
p_j: relative frequency of class j in T

- gini(T) is minimized if the classes in T are skewed ( )
- gini measures error rate of random classifier that assign classes to instances according to their prior frequencies.

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

→ If dataset D is split on A into two subsets D_1, D_2

Reduction of Impurity; $\Delta gini(A) = gini(D) - gini_A(D)$

- The attribute provides the smallest gini splitting D over A is chosen to split the node
  (largest reduction of impurity)
- Gini index is typically applied to produce binary splits

Ex:

Gini Index for Outlook Attribute;

dataset has 9 tuples labeled 'yes' 5 tuples labeled 'no'

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459 \rightarrow$$
$$gini(D) - gini(D_{rs}, D_o) = 0.459 - 0.357 = 0.102$$

{overcast, rainy} and {sunny}:

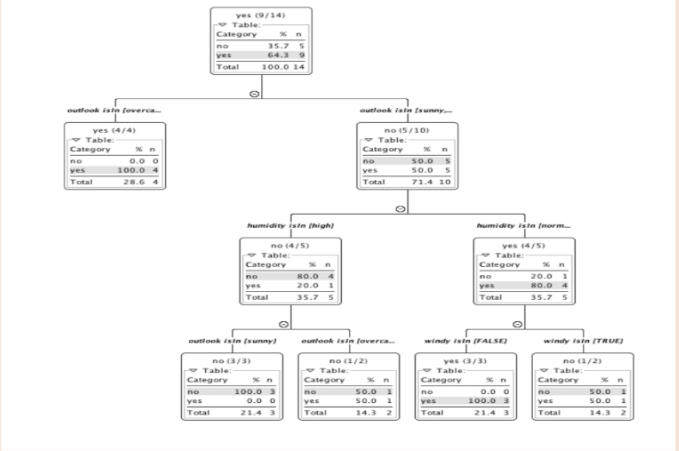
$$gini(D_{or}, D_s) = \frac{9}{14} gini([3,2]) + \frac{5}{14} gini([2,2]) = 0.394$$
(0.944, 0.480)

{rainy, sunny} and {overcast}:

$$gini(D_{rs}, D_o) = \frac{9}{14} gini([3,5]) + \frac{5}{14} gini([4,0]) = 0.357$$
(0.5, 0) → smallest gini ✓

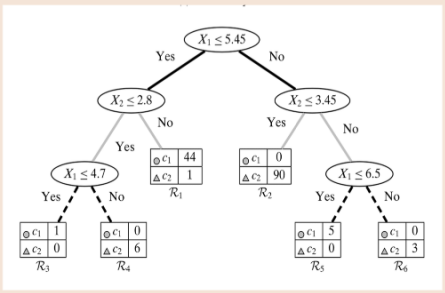
{Sunny, overcast} and {rainy}:

$$gini(D_{so}, D_r) = \frac{9}{14} gini([4,3]) + \frac{5}{14} gini([3,2]) = 0.457$$
(0.444, 0.480)



» Numerical Attributes

Outlook	Temperature	Humidity	Windy	Play
Sunny	85	85	False	no
Sunny	80	90	True	no
Overcast	83	86	False	yes
Rainy	70	96	False	yes
Rainy	68	80	False	yes
Rainy	65	70	True	no
Overcast	64	65	True	yes
Sunny	72	95	False	no
Sunny	69	70	False	yes
Rainy	75	80	False	yes
Sunny	75	70	True	yes
Overcast	72	90	True	yes
Overcast	81	75	False	yes
Rainy	71	91	True	no



- First, sort temperature values including class labels
- Then check all feasible cut points, choose the one with best IG

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

→ temp ≤ 70.5 (y:4 n:1)
temp > 70.5 (y:5 n:4)

info([4,1],[5,4]) = $\frac{5}{14}$ info([4,1]) + $\frac{9}{14}$ info([5,4]) = 0.894

IG (temperature at 70.5) = 1 − 0.894 = 0.045

» Avoid overfitting in Decision Trees

- Generated tree may overfit the training data → poor accuracy on unseen samples
- Too many branches, some may reflect anomalies due to the noise or outliers
- Two approaches to avoid overfitting:

1) Prepruning

- Stop tree construction early
- Do not split a node if this would result in the goodness measure falling below a threshold. (most popular: chi-squared test) → Difficult to choose threshold

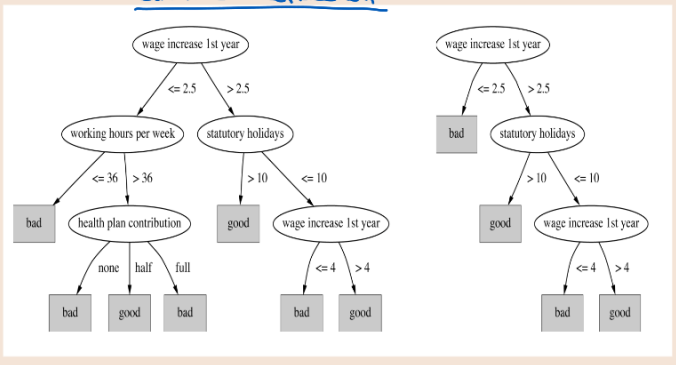
2) Post pruning

- First, build the tree, then prune it → sequence of pruned trees
- Use test set to decide which is the "best pruned tree"
- Two operation: 1) Subtree raising 2) Subtree replacement
- Possible strategies: 1) Error estimation 2) Significance testing 3) MDL Principle

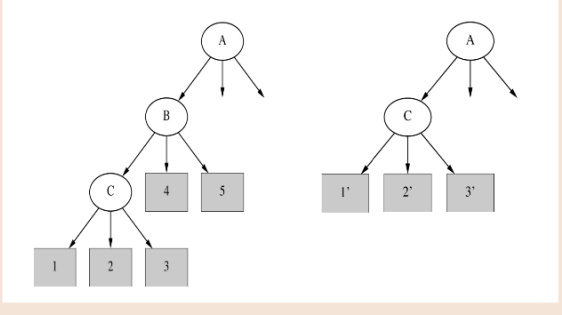
ALGORITHM 19.1. Decision Tree Algorithm

```
DECISIONTREE (D, η, π):
n ← |D| // partition size
n_i ← |{x_j | x_j ∈ D, y_j = c_i}| // size of class c_i
purity(D) ← max_i {n_i/n}
if n ≤ η or purity(D) ≥ π then // stopping condition
    c* ← arg max_{c_i} {n_i/n} // majority class
    create leaf node, and label it with class c*
    return
(split point*, score*) ← (∅, 0) // initialize best split point
foreach (attribute X_j) do
    if (X_j is numeric) then
        (v, score) ← EVALUATE-NUMERIC-ATTRIBUTE(D, X_j)
        if score > score* then (split point*, score*) ← (X_j ≤ v, score)
    else if (X_j is categorical) then
        (V, score) ← EVALUATE-CATEGORICAL-ATTRIBUTE(D, X_j)
        if score > score* then (split point*, score*) ← (X_j ∈ V, score)
// partition D into D_Y and D_N using split point*, and call recursively
D_Y ← {x ∈ D | x satisfies split point*}
D_N ← {x ∈ D | x does not satisfy split point*}
create internal node split point*, with two child nodes, D_Y and D_N
DECISIONTREE(D_Y); DECISIONTREE(D_N)
```

Subtree Replacement



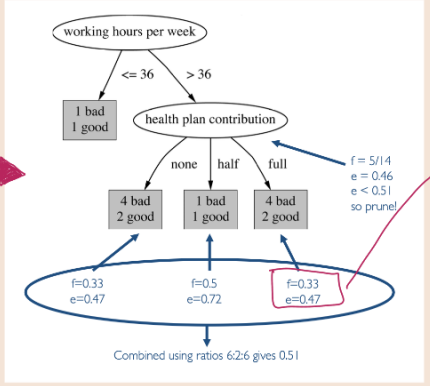
Subtree Raising



- **Estimating Error Rates**
  - Prune only if it reduces the estimation error.
  - Example: C4.5's method

$$e = \left(f + \frac{z^2}{2N} + z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}}\right) / \left(1 + \frac{z^2}{N}\right)$$

upper bound for error estimation for a node
f: given error on training data
N: # of instances covered by the leaf
if c = 7.25 → z = 0.69 (from normal dist)



$$e = \frac{0.33 + \frac{0.69^2}{2\cdot 6} + 0.69\sqrt{\frac{0.33}{6} - \frac{0.33^2}{6} + \frac{0.69^2}{4\cdot 6^2}}}{1 + \frac{0.69^2}{6}} \approx 0.47$$

» Regression Trees

- Decision trees can also be used to predict the value of a numerical target value
- They search for the best split that minimizes an impurity measure
  - measured as expected error reduction or SDR (Standard dev. reduction)

$$SDR = \sigma(D) - \sum_i \frac{|D_i|}{|D|} \cdot \sigma(D_i)$$

→ D: original data, D_i: partitions, σ: std. dev of target