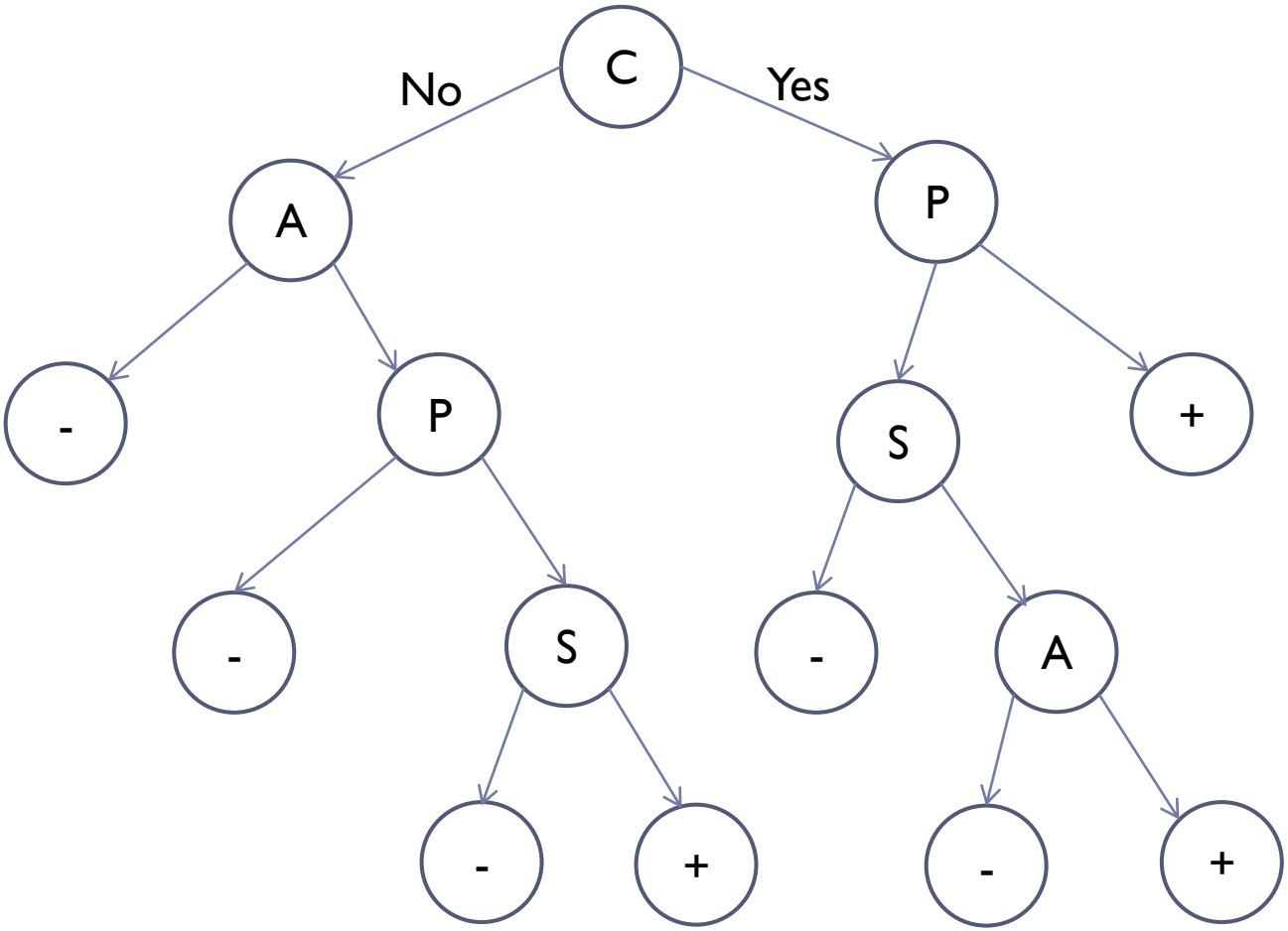
Decision tree

- One of the most intuitive classifiers that is easy to understand and construct
  - However, it also works very (very) well
- Categorical features are preferred. If feature values are continuous, they are discretized first.
- Application: Database mining

Example

- Attributes:
  - A: age>40
  - C: chest pain
  - S: smoking
  - P: physical test

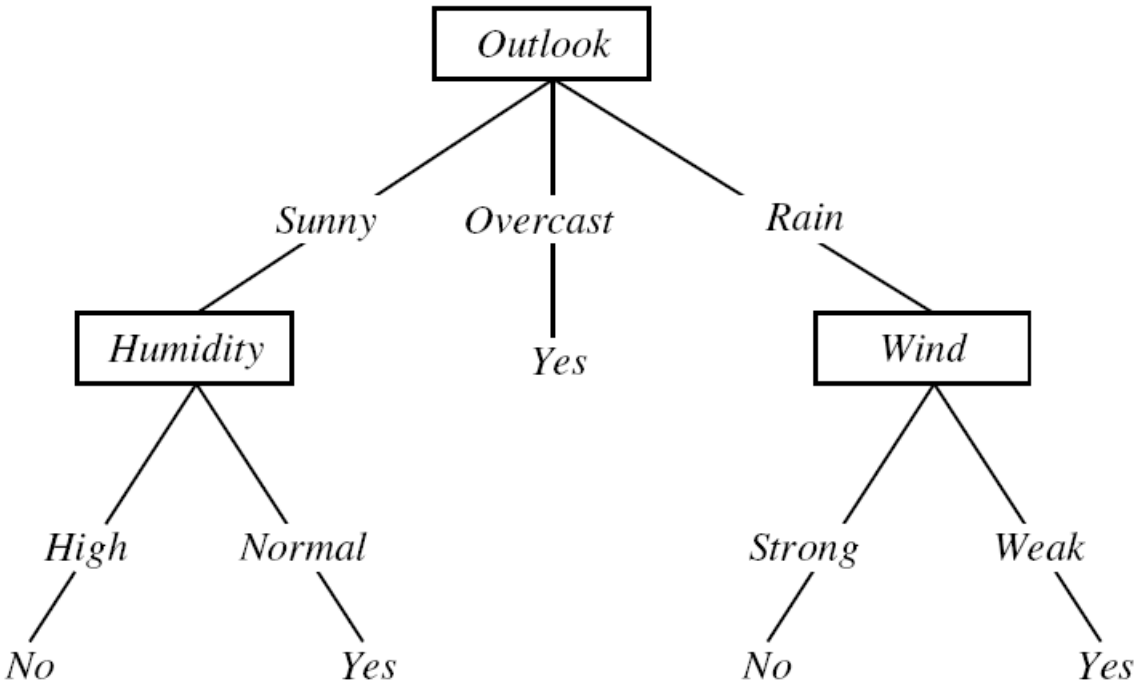


- Label:
  - Heart disease (+), No heart disease (-)

Decision tree: structure

- Leaves (terminal nodes) represent target variable
  - Each leaf represents a class label

- Each internal node denotes a test on an attribute
  - Edges to children for each of the possible values of that attribute



Day	Outlook	Temperature	Humidity	Wind	PlayTenı
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Decision tree: learning

- Decision tree learning: construction of a decision tree from training samples.
  - Decision trees used in data mining are usually classification trees
- There are many specific decision-tree learning algorithms, such as:
  - ID3
  - C4.5
- Approximates functions of usually discrete domain
  - The learned function is represented by a decision tree

Decision tree learning

- Learning an optimal decision tree is NP-Complete
  - Instead, we use a greedy search based on a heuristic
  - We cannot guarantee to return the globally-optimal decision tree.

- The most common strategy for DT learning is a greedy top-down approach
  - chooses a variable at each step that best splits the set of items.

- Tree is constructed by splitting samples into subsets based on an attribute value test in a recursive manner

How to construct basic decision tree?

- We prefer decisions leading to a simple, compact tree with few nodes

- Which attribute at the root?
  - Measure: how well the attributes split the set into homogeneous subsets (having same value of target)
    - Homogeneity of the target variable within the subsets.

- How to form descendant?
  - Descendant is created for each possible value of A
    - Training examples are sorted to descendant nodes

Constructing a decision tree

```
Function FindTree(S,A)
   If empty(A) or all labels of the samples in S are the same
        status = leaf
        class = most common class in the labels of S
   else
        status = internal
        a ←bestAttribute(S,A)
        LeftNode = FindTree(S(a=1), A \ {a})
        RightNode = FindTree(S(a=0), A \ {a})
   end
end
```

S: samples, A: attributes

Recursive calls to create left and right subtrees
S(a=1) is the set of samples in S for which a=1

Top down, Greedy, No backtrack

Constructing a decision tree

```
Function FindTree(S,A)
   If empty(A) or all labels of the samples in S are the same
        status = leaf
        class = most common class in the labels of S
   else
        status = internal
        a ←bestAttribute(S,A)
        LeftNode = FindTree(S(a=1),A \ {a})
        RightNod
   end
end
```

S: samples, A: attributes

Tree is constructed by splitting samples into subsets based on an attribute value test in a recursive manner

- The recursion is completed when all members of the subset at a node have the same label
- or when splitting no longer adds value to the predictions

ID3

ID3 (Examples, Target_Attribute, Attributes)

Create a root node for the tree

If all examples are positive, return the single-node tree Root, with label = +

If all examples are negative, return the single-node tree Root, with label = -

If number of predicting attributes is empty then

return Root, with label = most common value of the target attribute in the examples

else

A = The Attribute that best classifies examples.

Testing attribute for Root = A.

for each possible value, v_i, of A

Add a new tree branch below Root, corresponding to the test A =v_i .

Let Examples(v_i) be the subset of examples that have the value for A

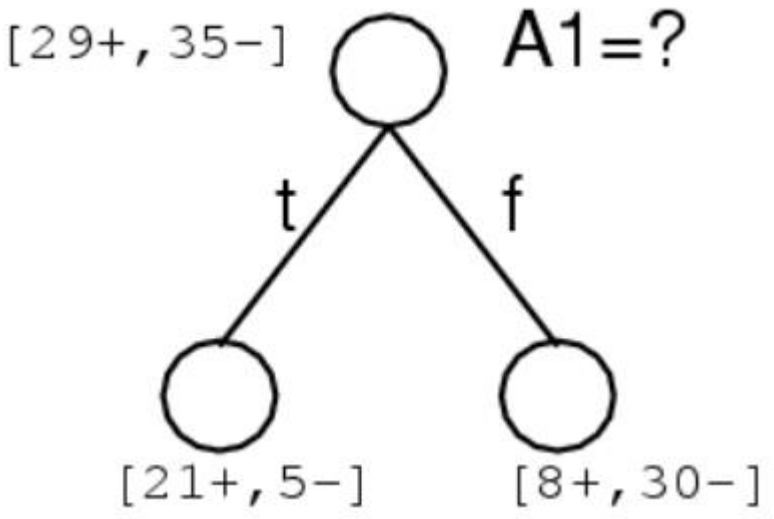
if Examples(v_i) is empty then

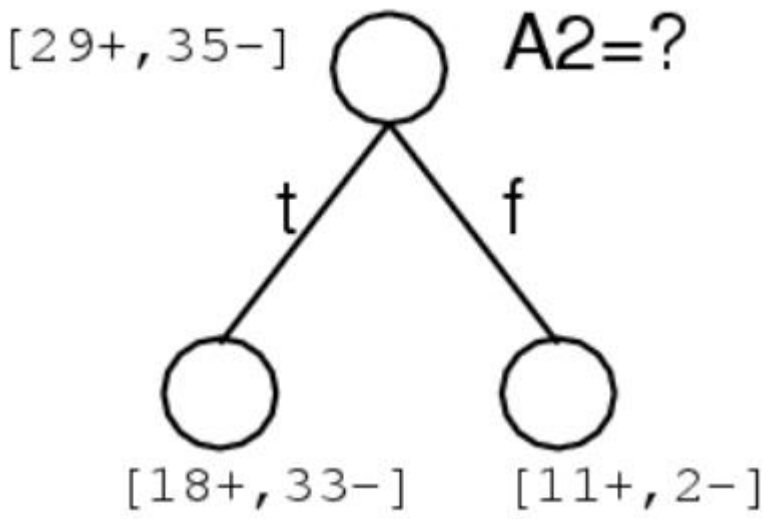
below this new branch add a leaf node with label = most common target value in the examples

else below this new branch add subtree **ID3 (Examples(v_i), Target_Attribute, Attributes – {A})**

return Root

Which attribute is the best?





Which attribute is the best?

- A variety of heuristics for picking a good test
  - Information gain: originated with ID3 (Quinlan,1979).
  - Gini impurity
  - …
- These metrics are applied to each candidate subset, and the resulting values are combined (e.g., averaged) to provide a measure of the quality of the split.

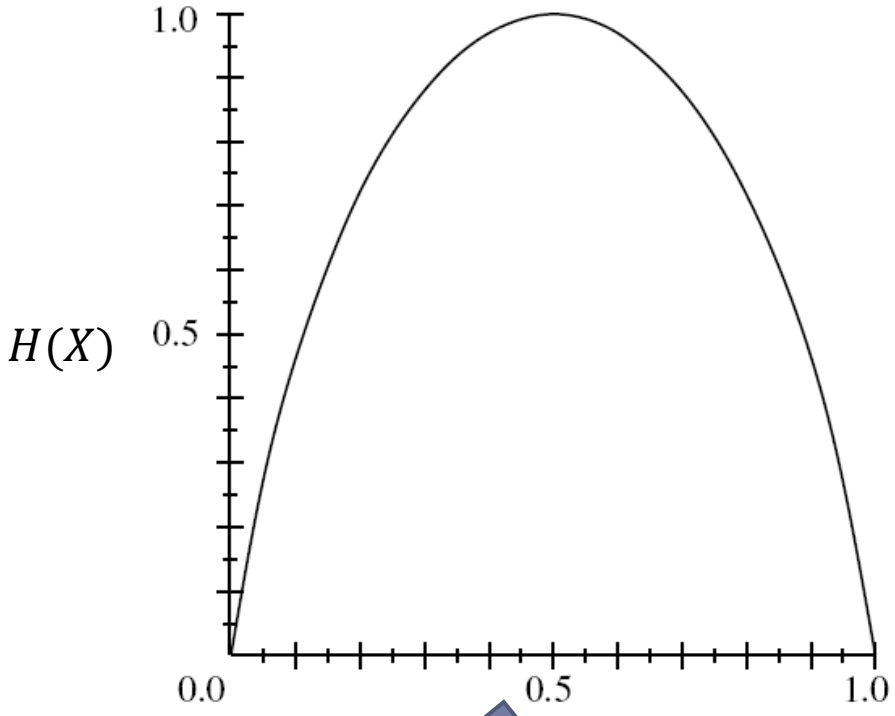
Entropy

$$H(X) = -\sum_{x_i \in X} P(x_i) \log P(x_i)$$

- Entropy measures the uncertainty in a specific distribution

- Information theory:
  - $H(X)$: expected number of bits needed to encode a randomly drawn value of X (under most efficient code)
    - Most efficient code assigns $-\log P(X = i)$ bits to encode $X = i$
    - $\Rightarrow$ expected number of bits to code one random X is $H(X)$

Entropy for a Boolean variable



Entropy as a measure of impurity

$$H(X) = -0.5 \log_2 \frac{1}{2} - 0.5 \log_2 \frac{1}{2} = 1$$

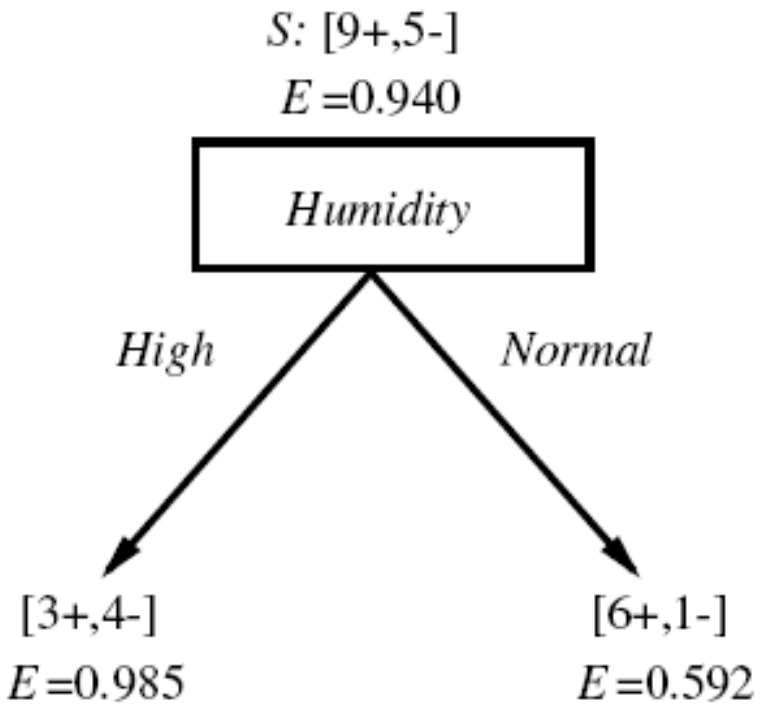
$$H(X) = -1 \log_2 1 - 0 \log_2 0 = 0$$

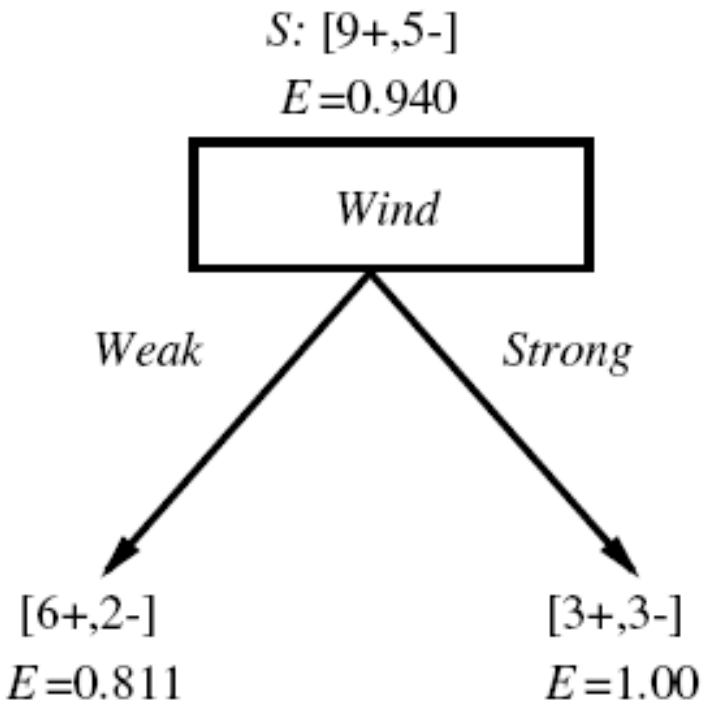
Information Gain (IG)

$$Gain(S, A) \equiv H_S(Y) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H_{S_v}(Y)$$

- A: variable used to split samples
- Y: target variable
- S: samples

Information Gain: Example



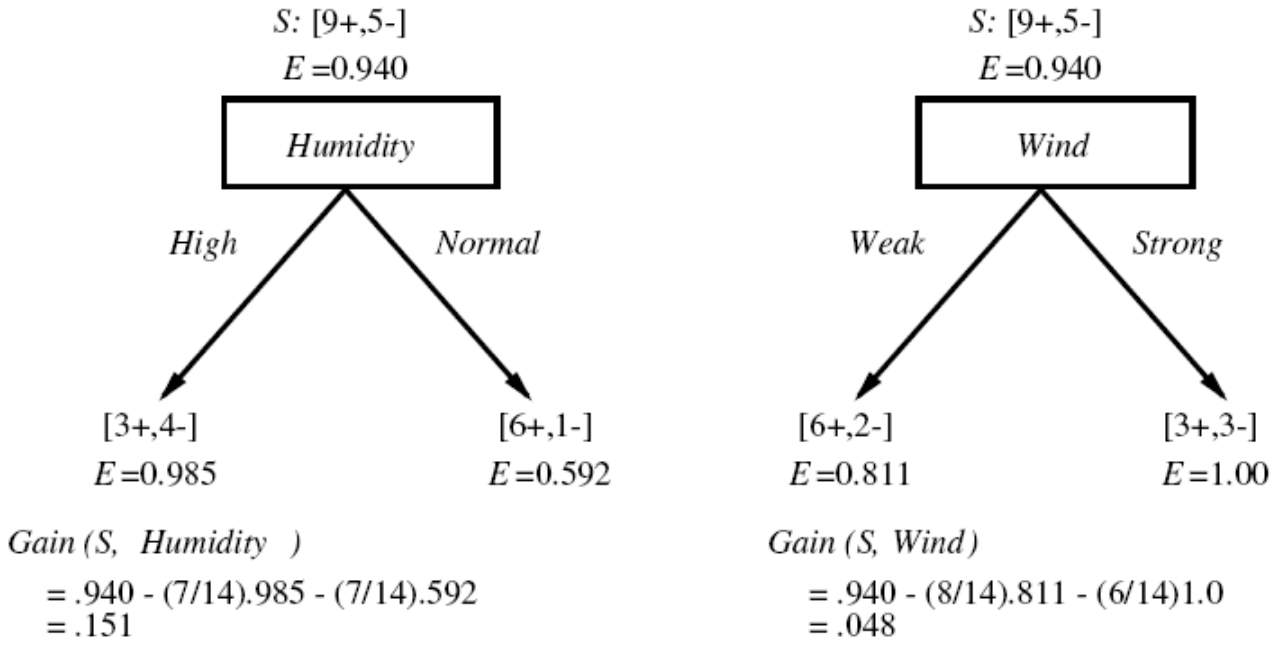


Conditional entropy: example

- $H(Y|Humidity)$

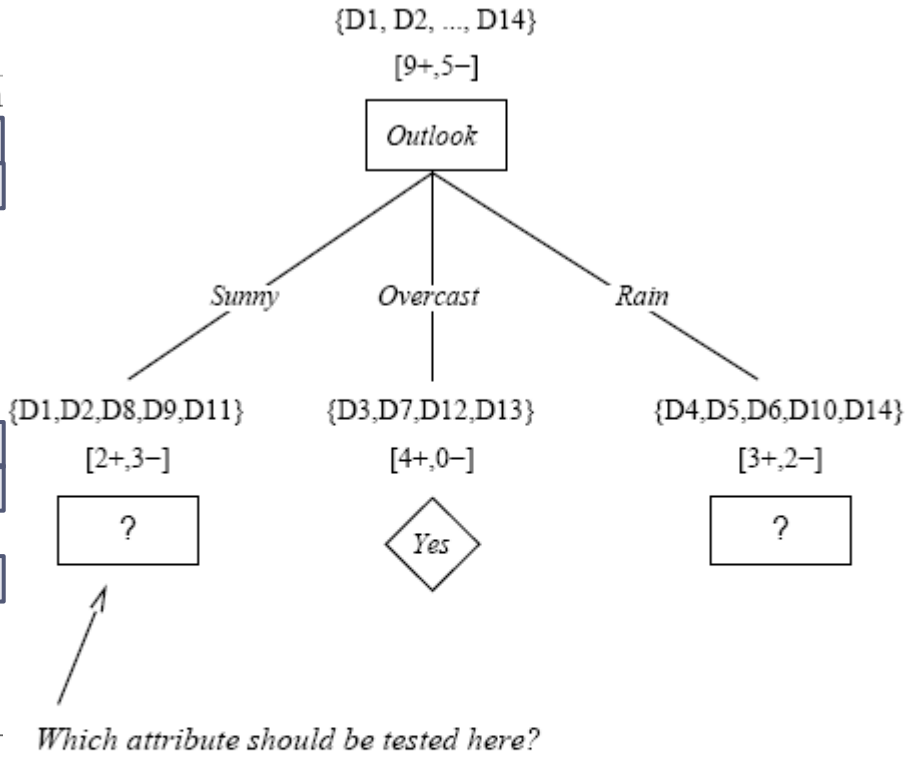
  $= \frac{7}{14} \times H(Y|Humidity = High) + \frac{7}{14} \times H(Y|Humidity = Normal)$
- $H(Y|Wind)$

  $= \frac{8}{14} \times H(Y|Wind = Weak) + \frac{6}{14} \times H(Y|Wind = Strong)$



Information Gain: Example

Day	Outlook	Temperature	Humidity	Wind	PlayTen
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



S_{sunny} = {D1,D2,D8,D9,D11}

Gain (S_{sunny}, *Humidity*) = .970 − (3/5) 0.0 − (2/5) 0.0 = .970

Gain (S_{sunny}, *Temperature*) = .970 − (2/5) 0.0 − (2/5) 1.0 − (1/5) 0.0 = .570

Gain (S_{sunny}, *Wind*) = .970 − (2/5) 1.0 − (3/5) .918 = .019

How to find the best attribute?

- Information gain as our criteria for a good split
  - attribute that maximizes information gain is selected

- When a set of S samples have been sorted to a node, choose j-th attribute for test in this node where:

$$j = \underset{i \in \text{remaining atts.}}{\text{argmax}} \; Gain(S, X_i)$$

$$= \underset{i \in \text{remaining atts.}}{\text{argmax}} \; H_S(Y) - H_S(Y|X_i)$$

$$= \underset{i \in \text{remaining atts.}}{\text{argmin}} \; H_S(Y|X_i)$$

ID3 algorithm: Properties

- The algorithm
  - either reaches homogenous nodes
  - or runs out of attributes

- Guaranteed to find a tree consistent with any conflict-free training set
  - ID3 hypothesis space of all DTs contains all discrete-valued functions
  - Conflict free training set: identical feature vectors always assigned the same class

- But not necessarily find the simplest tree (containing minimum number of nodes).
  - a greedy algorithm with locally-optimal decisions at each node (no backtrack).

Decision tree learning: Function approximation problem

- **Problem Setting**:
  - Set of possible instances X
  - Unknown target function $f: X \rightarrow Y$ (Y is discrete valued)
  - Set of function hypotheses $H = \{ h \mid h : X \rightarrow Y \}$
    - h is a DT where tree sorts each $\boldsymbol{x}$ to a leaf which assigns a label y
- **Input**:
  - Training examples $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}$ of unknown target function f
- **Output**:
  - Hypothesis $h \in H$ that best approximates target function f

Decision tree hypothesis space

- Suppose attributes are Boolean
- Disjunction of conjunctions
- Which trees to show the following functions?
  - $y = x_1 \; and \; x_5$
  - $y = x_1 \; or \; x_4$
  - $y = (x_1 \; and \; x_5) \; or (x_2 \; and \; \neg x_4)$?

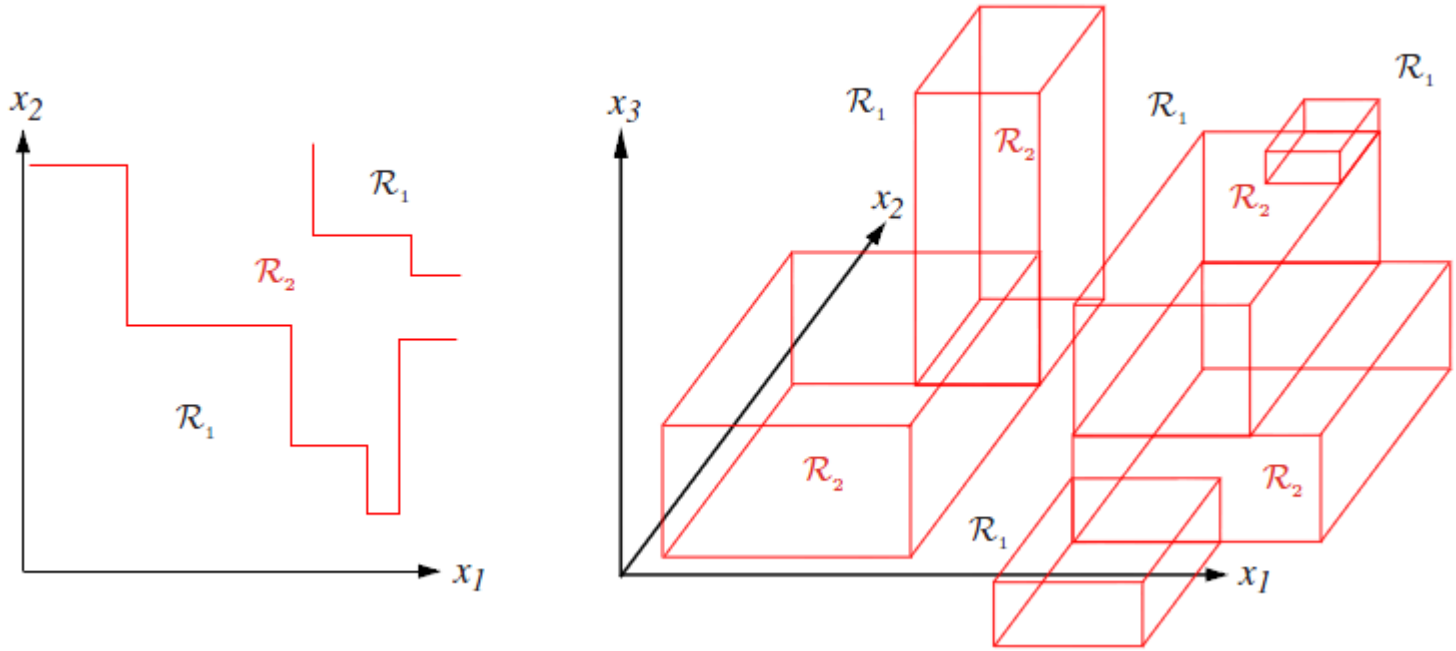
Decision tree as a rule base

- Decision tree = a set of rules

- Disjunctions of conjunctions on the attribute values
  - Each path from root to a leaf = conjunction of attribute tests
  - All of the leafs with $y = i$ are considered to find the rule for $y = i$

How partition instance space?

- Decision tree
  - Partition the instance space into axis-parallel regions, labeled with class value



[Duda & Hurt 's Book]

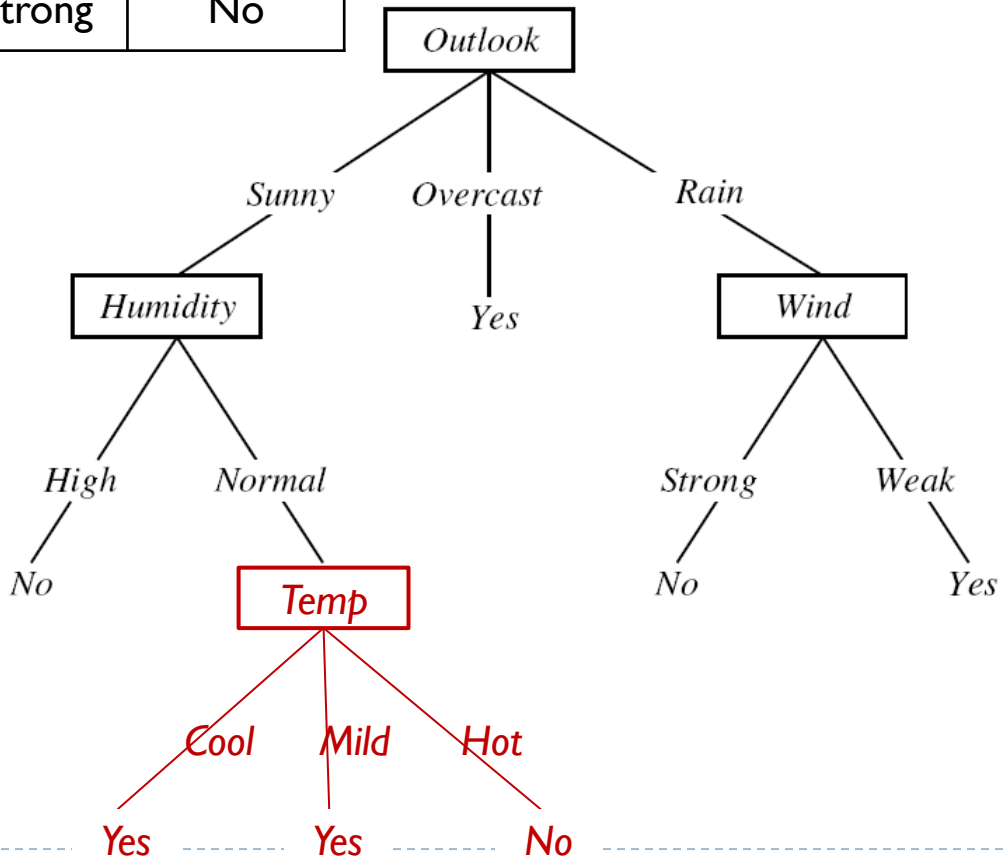
Over-fitting problem

- ID3 perfectly classifies training data (for consistent data)
  - It tries to memorize every training data
  - Poor decisions when very little data (it may not reflect reliable trends)
    - Noise in the training data: the tree is erroneously fitting.
    - A node that "should" be pure but had a single (or few) exception(s)?
- For many (non relevant) attributes, the algorithm will continue to split nodes
  - leads to over-fitting!

Over-fitting problem: an example

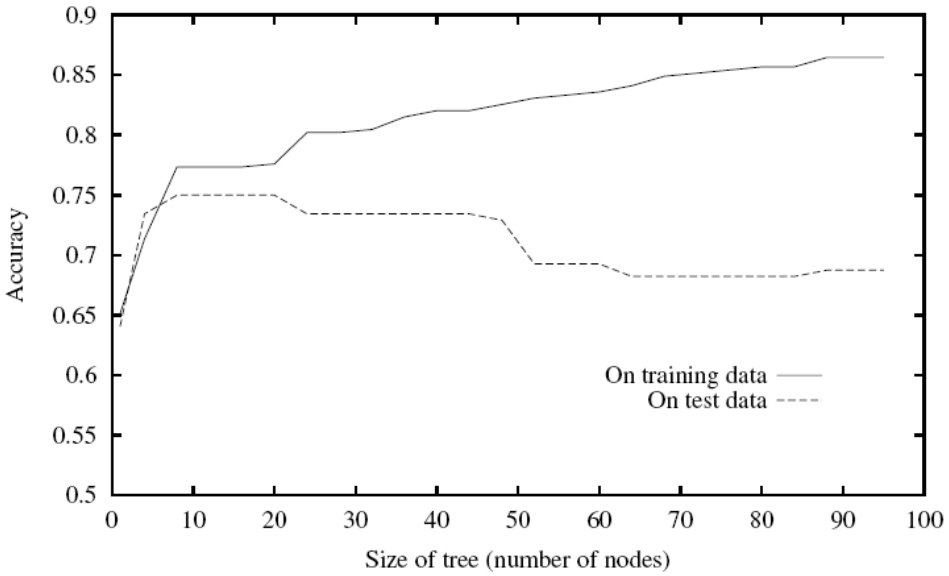
- Consider adding a (noisy) training example:

Outlook	Temp	Humidity	Wind	PlayTennis
Sunny	Hot	Normal	Strong	No



Over-fitting in decision tree learning

- Hypothesis space H: decision trees
- Training (emprical) error of $h \in H$: $error_{train}(h)$
- Expected error of $h \in H$: $error_{true}(h)$
- h overfits training data if there is a $h' \in H$ such that
  - $error_{train}(h) < error_{train}(h')$
  - $error_{true}(h) > error_{true}(h')$



A question?

- How can it be made smaller and simpler?
  - Early stopping
    - When should a node be declared as a leaf?
    - If a leaf node is impure, how should the category label be assigned?
  - Pruning?
    - Build a full tree and then post-process it

Avoiding overfitting

1) **Stop growing** when the data split is not statistically significant.

2) Grow full tree and then **prune** it.
   - More successful than stop growing in practice.

3) How to select "best" tree:
   - Measure performance over separate validation set
   - MDL: minimize

$$size(tree) + size(missclassifications(tree))$$

Reduced-error pruning

- Split data into train and validation set
- Build tree using training set
- Do until further pruning is harmful:
  - Evaluate impact on validation set when pruning sub-tree rooted at each node
    - Temporarily remove sub-tree rooted at node
    - Replace it with a leaf labeled with the current majority class at that node
    - Measure and record error on validation set
  - Greedily remove the one that most improves validation set accuracy (if any).

Produces smallest version of the most accurate sub-tree.

C4.5

- C4.5 is an extension of ID3
  - Learn the decision tree from samples (allows overfitting)
  - Convert the tree into the equivalent set of rules
  - Prune (generalize) each rule by removing any precondition that results in improving estimated accuracy
  - Sort the pruned rules by their estimated accuracy
    - consider them in sequence when classifying new instances

- Why converting the decision tree to rules before pruning?
  - Distinguishing among different contexts in which a decision node is used
  - Removes the distinction between attribute tests that occur near the root and those that occur near the leaves

Overfitting

- Decision Trees make very few assumptions about the training data!
- If left unconstrained, their structure will adapt itself to the training data, fitting it very closely, and most likely overfitting it!
- To avoid overfitting the training data, you need to restrict the Decision Tree's freedom during training.
- Which parameters can be restricted?
  - Max Tree Depth: Maximum depth of the tree.
  - Max Leaf Nodes: Maximum number of leaf nodes.
  - Min Sample Leaf: Minimum number of samples a leaf node must have.
  - Min Sample Split: Minimum number of samples a node must have before it can be split.
  - Max Features: Maximum number of features that are evaluated for splitting at each node)

Continuous attributes

- Tests on continuous variables as boolean?
- Either use threshold to turn into binary or discretize
- It's possible to compute information gain for all possible thresholds (there are a finite number of training samples)

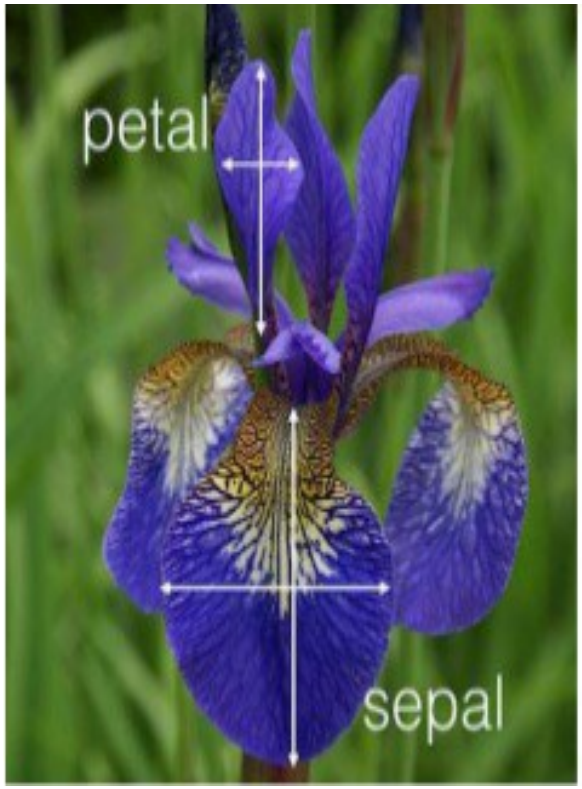
Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

- Harder if we wish to assign more than two values (can be done recursively)

CART Algorithm

- The algorithm first splits the training set in two subsets using a single feature k and a threshold t_k.
- How does it choose k and t_k?
  It searches for the pair (k, t_k) producing the purest subsets, which are weighted by their size.
- Once it has successfully split the training set in two, it splits the subsets using the same logic, then the sub-subsets and so on, recursively.
- CART algorithm is a greedy algorithm: it greedily searches for an optimum split at the top level, then repeats the process at each level.
- It does not check whether or not the split will lead to the lowest possible impurity several levels down.

Decision Trees in Practice



Training / test data

Features				Labels
Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	Iris setosa
4.9	3.0	1.4	0.2	Iris setosa
7.0	3.2	4.7	1.4	Iris versicolor
6.4	3.2	4.5	1.5	Iris versicolor
6.3	3.3	6.0	2.5	Iris virginica
5.8	3.3	6.0	2.5	Iris virginica

Figure: Iris Dataset, Source

Decision Trees in Practice

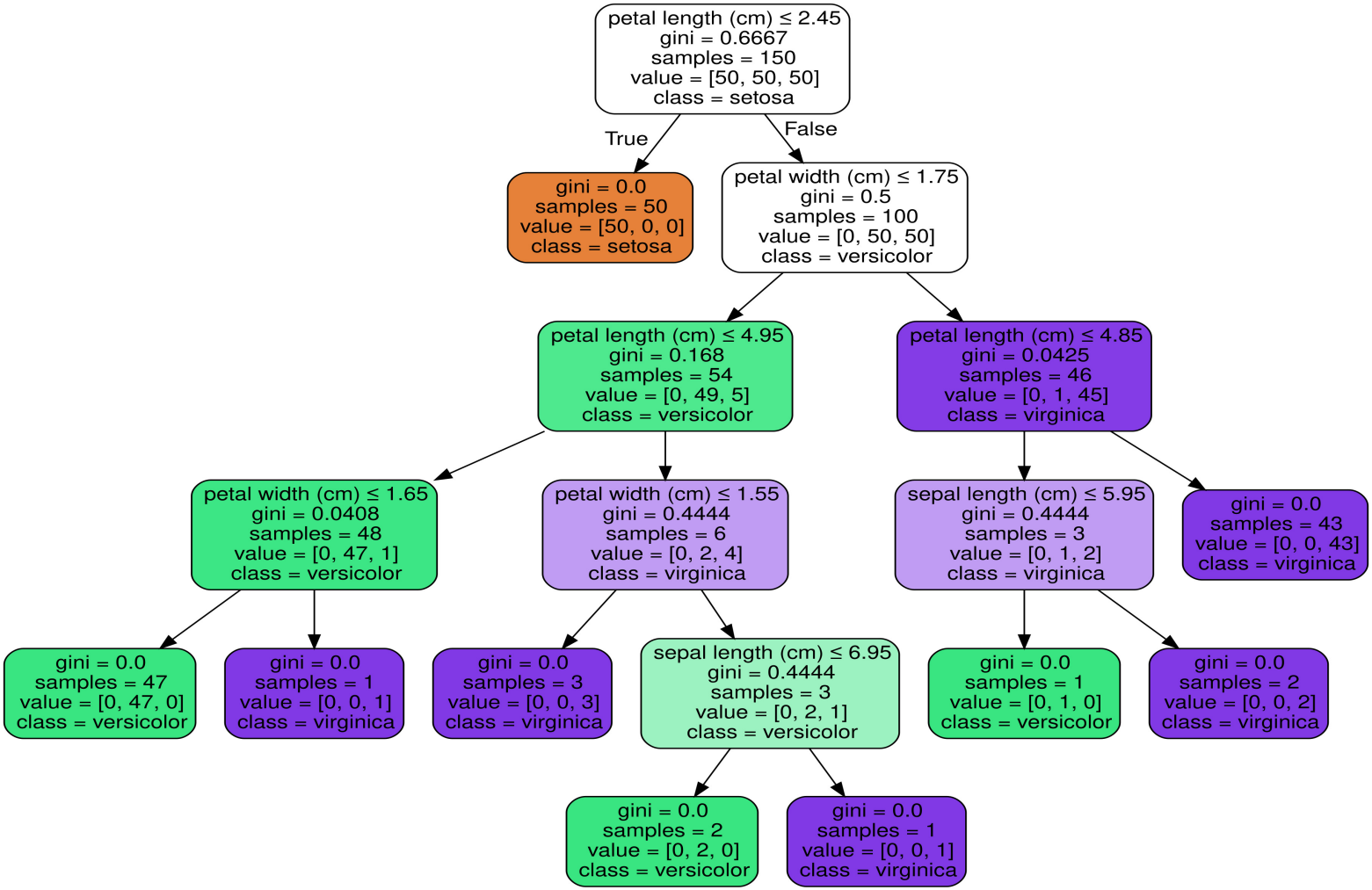


Figure: Decision Trees on Iris Dataset, Source

Other splitting criteria

- Information gain are biased in favor of those attributes with more levels.
  - More complex measures to select attribute

- Example: attribute Date

- Gain Ratio:

$$GainRatio(S,A) \equiv \frac{Gain(S,A)}{-\sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \log \frac{|S_v|}{|S|}}$$

مثالی از نحوه محاسبه شاخص نسبت بهره (Gain Ratio)

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- محاسبه برای ویژگی income:
  - از قبل داشتیم:

$$Gain(income) = 0.029$$

$$SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2(\frac{4}{14}) - \frac{6}{14} \times \log_2(\frac{6}{14}) - \frac{4}{14} \times \log_2(\frac{4}{14}) = 0.926$$

GainRatio(income) = 0.029/0.926 = 0.031

شاخص جینی (Gini Index)

- اگر مجموعه داده D، شامل نمونه‌هایی از n کلاس مختلف باشد: $gini(D) = 1 - \sum_{j=1}^{n} p_j^2$
  - که در آن، p_j برابر با فرکانس نسبی عناصر کلاس j در D است.
- اگر ویژگی A، داده D را به دو بخش D_1 و D_2 تفکیک کند، مقدار جینی D عبارتست از:

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

- در اینصورت، میزان کاهش ناخالصی، برابر است با: $\Delta gini(A) = gini(D) - gini_A(D)$
- ویژگی با کمینه شاخص جینی (بیشینه میزان کاهش ناخالصی)، جهت انشعاب، تعیین می‌شود.

نمونه محاسبه شاخص جینی (Gini Index)

به ازای داده زیر، ۹ نمونه مثبت و پنج نمونه منفی داریم، لذا:

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

نمونه محاسبه شاخص جینی (ادامه)

- فرض کنید ویژگی "income"، داده را به دو بخش شامل ۱۰ نمونه در D_1: {low, medium} و ۴ نمونه در D_2: {high}، تفکیک کند:

$$gini_{income \in \{low, medium\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_2) = 0.443$$

- به همین ترتیب:

$$\text{Gini}_{\{low,high\}} = 0.458; \text{Gini}_{\{medium,high\}} = 0.450$$

- لذا انشعاب بصورت دو بخش {low, medium}و {high}، کمترین میزان شاخص جینی را بدست می‌دهد.

مقایسه شاخص‌های انتخاب ویژگی

- هر سه شاخص، عملکرد نسبتاً مطلوبی دارند، در عین حال:
  - بهره اطلاعات (Information gain): به طرف ویژگی‌های چند مقداره، بایاس دارد.
  - نسبت بهره (Gain ratio): گرایش بسوی انجام انشعاباتی دارد که طی آنها، یک بخش، بسیار کوچکتر از بقیه باشد.
  - شاخص جینی (Gini index):
    - به طرف ویژگی‌های چند مقداره، بایاس دارد.
    - منجر به تولید بخش‌بندیهایی با اندازه نسبتاً برابر دارد که هر کدام، نسبتاً خالص باشند.
- انتخاب شاخص مناسب بنا به شرایط مساله و داده‌های در اختیار

Decision Trees for Regression

- It is the same as classification with a few subtle modifications:
  - The main difference is that instead of predicting a class in each node, it predicts a value.
  - The prediction for each node is simply the average target value of the all training instances associated to this leaf node.
  - Instead of trying to split the training set in a way that minimizes impurity, CART algorithm now tries to split the training set in a way that minimizes the MSE.
  - The MSE at a given node is also often referred to as intra-node variance, and the splitting criterion is thus called variance reduction.
  - Likewise to classification tasks, decision trees are prone to overfitting when dealing with regression tasks.

Positiveness of Information Gain

```
------
Data Set Characteristics:

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive

    :Median Value (attribute 14) is usually the target

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's
```

Figure: Boston Dataset, Source

Decision Trees for Regression

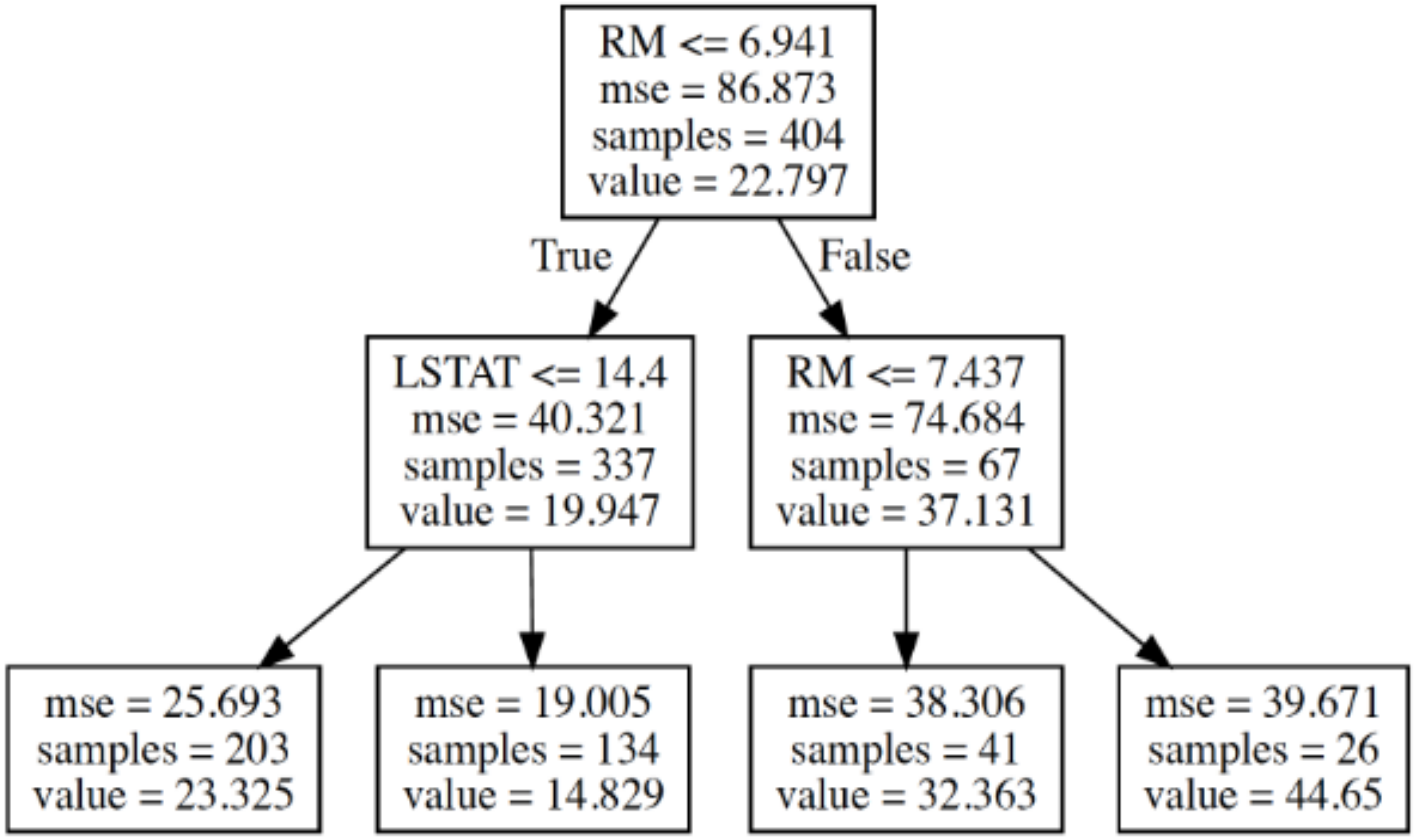


Figure: Concavity of Entropy, Source

Decision tree advantages

- Simple to understand and interpret
- Requires little data preparation and also can handle both numerical and categorical data
- Time efficiency of learning decision tree classifier
  - Cab be used on large datasets
- Robust: Performs well even if its assumptions are somewhat violated