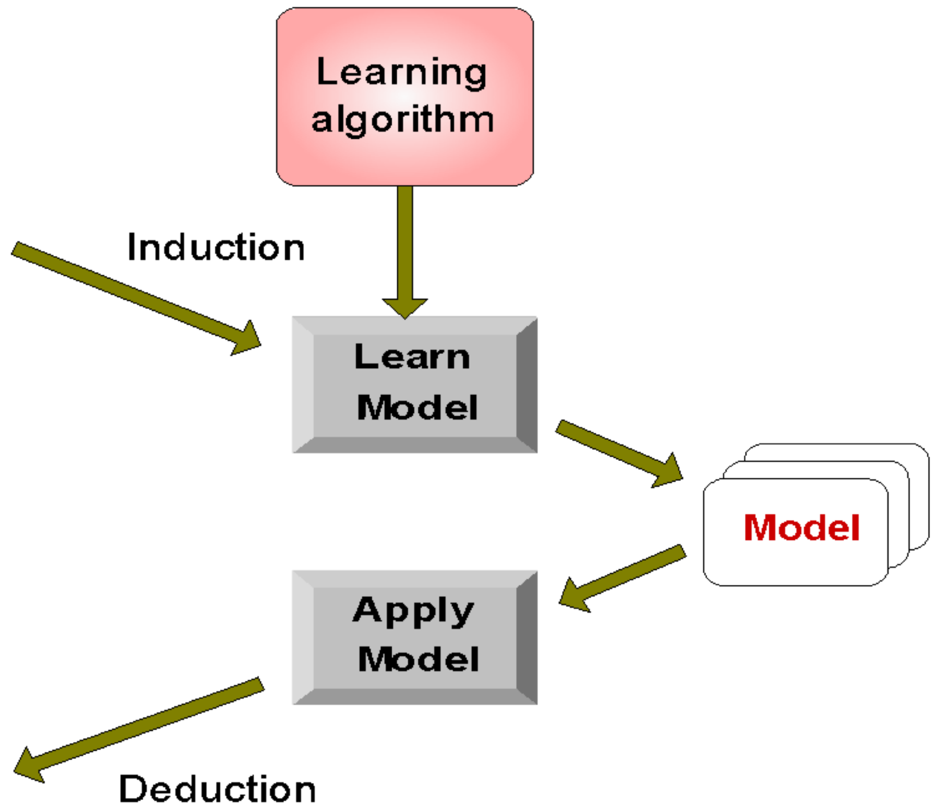
Regression Trees

Slides from
Stephen Marsland and
Longin Jan Latecki

Illustrating Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



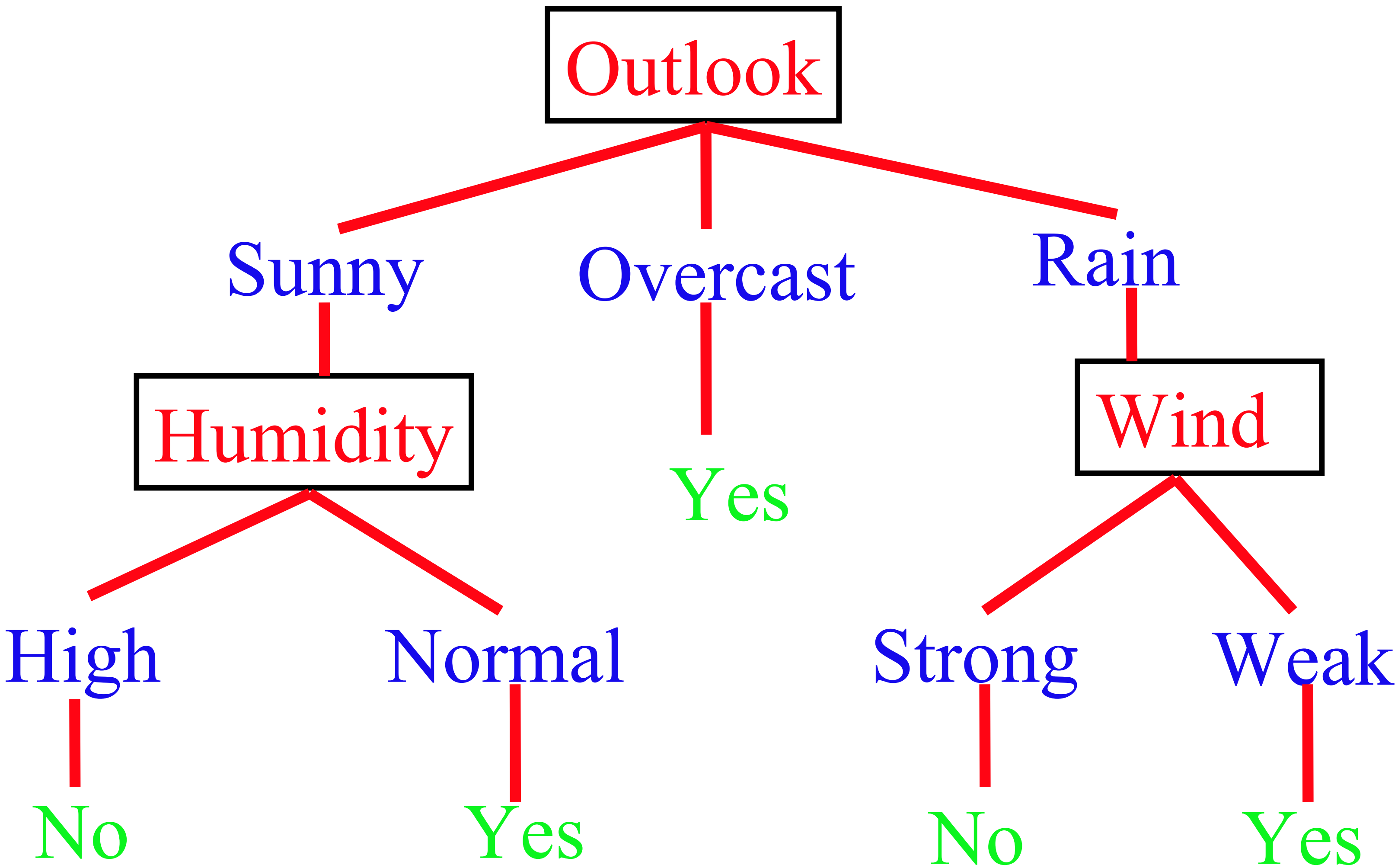
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Decision Trees

- Split classification down into a series of choices about features in turn
- Lay them out in a tree
- Progress down the tree to the leaves

Play Tennis Example



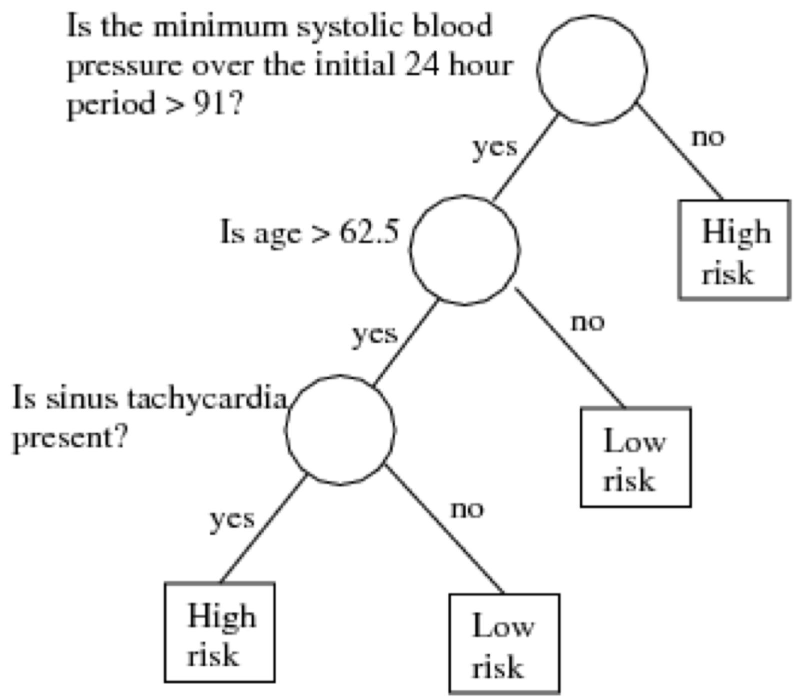
Day	Outlook	Temp	Humid	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

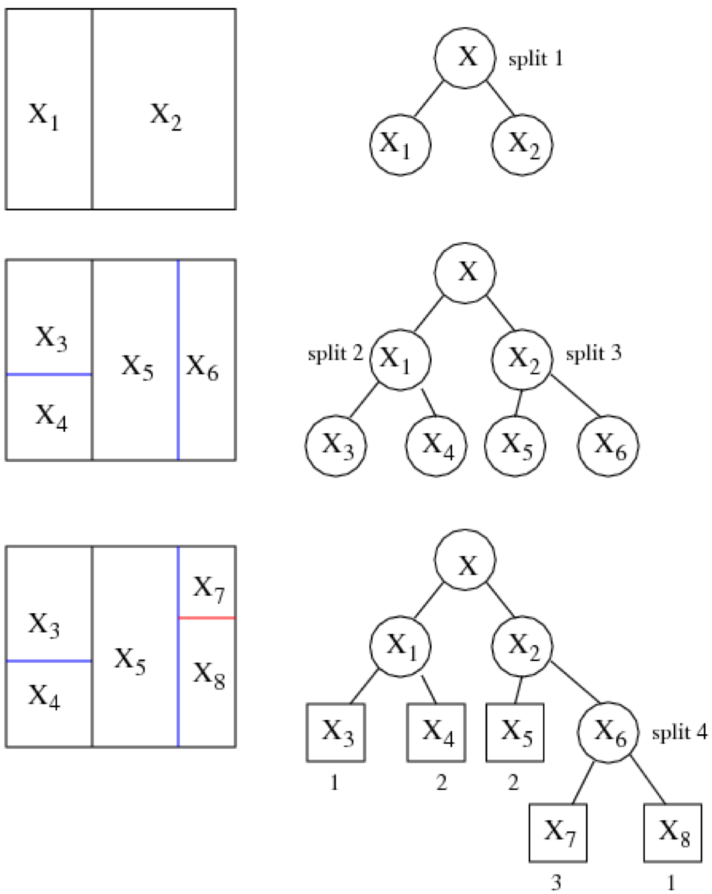
Rules and Decision Trees

- Can turn the tree into a set of rules:
  - (outlook = sunny & humidity = normal) |
    (outlook = overcast) |
    (outlook = rain & wind = weak)

- How do we generate the trees?
  - Need to choose features
  - Need to choose order of features

A tree structure classification rule for a medical example



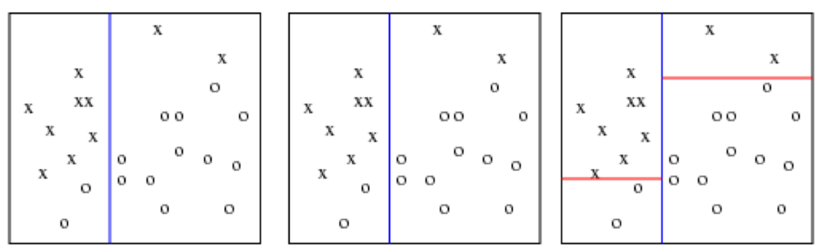


The construction of a tree involves the following three elements:

1. The selection of the splits.
2. The decisions when to declare a node as terminal or to continue splitting.
3. The assignment of each terminal node to one of the classes.

Goodness of split

The goodness of split is measured by an **impurity** function defined for each node.
Intuitively, we want each leaf node to be "pure", that is, one class dominates.



How to determine the Best Split

- Greedy approach:
  - Nodes with homogeneous class distribution are preferred
- Need a measure of node impurity:

C0: 5
C1: 5

Non-homogeneous,

High degree of impurity

C0: 9
C1: 1

Homogeneous,

Low degree of impurity

Measures of Node Impurity

Entropy

Gini Index

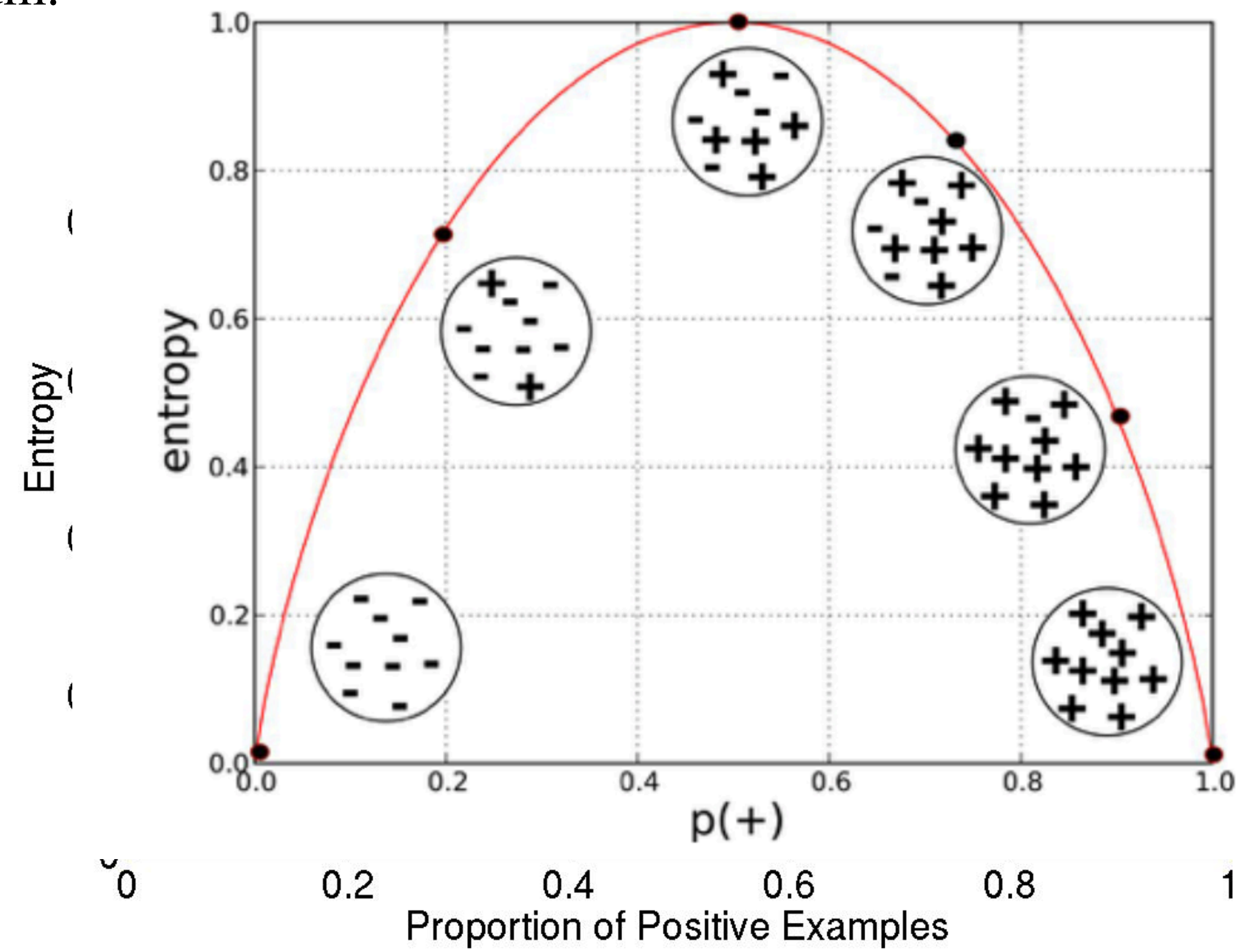
Entropy

$$H(p) = \sum_i p_i \log_2 p_i$$

- Let F be a feature with possible values $f_1, \ldots, f_n$
- Let p be a pdf (probability density function) of F; usually p is simply given by a histogram $(p_1, \ldots, p_n)$, where p_i is the proportion of the data that has value $F=f_i$.
- Entropy of p tells us how much extra information we get from knowing the value of the feature, i.e, $F=f_i$ for a given data point .
- Measures the amount in impurity in the set of features
- Makes sense to pick the features that provides the most information

Entropy for a distribution over a binary variable

E.g., if F is a feature with two possible values 0 and 1, and p_0=1 and p_1=0, then we get no new information from knowing that F=+1 for a given example. Thus the entropy is zero. If p1=0.5 and p2=0.5, then the entropy is at maximum.



Entropy and Decision Tree

Entropy at a given node t:

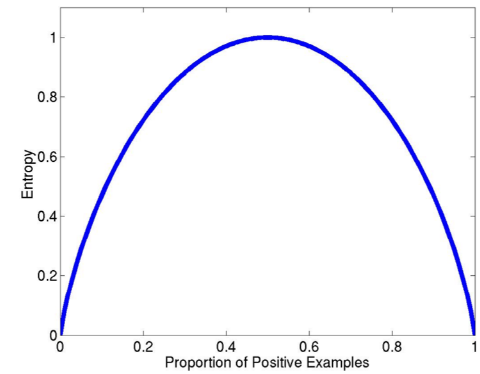
$$\boxed{Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)}$$

(NOTE: *p(j | t)* is the relative frequency of class j at node t).

- Measures homogeneity of a node.
  - Maximum ($\log n_c$) when records are equally distributed among all classes implying least information
  - Minimum (0.0) when all records belong to one class, implying most information
- Entropy based computations are similar to the GINI index computations

Examples for computing Entropy

$$\boxed{Entropy(t) = -\sum_j p(j\,|\,t)\log_2 p(j\,|\,t)}$$



C1	**0**
C2	**6**

P(C1) = 0/6 = 0   P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6   P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

C1	**2**
C2	**4**

P(C1) = 2/6   P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Splitting Based on Information Gain

Information Gain:

$$\boxed{GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)}$$

Parent Node, p is split into k partitions;

n_i is number of records in partition I

Measures Reduction in Entropy achieved because of the split.

Choose the split that achieves most reduction (maximizes GAIN)

Used in ID3 and C4.5

Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

Information Gain

$$\text{Gain}(S, F) = \text{Entropy}(S) - \sum_{f \in values(F)} \frac{|S_f|}{|S|} \text{Entropy}(S_f)$$

- Choose the feature that provides the highest information gain over all examples
- That is all there is to ID3:
  - At each stage, pick the feature with the highest information gain

Example

- Values(Wind) = Weak, Strong
- S = [9+, 5-]
- S(Weak) <- [6+, 2-]
- S(Strong) <- [3+, 3-]
- Gain(S,Wind) =
  - Entropy(S) -
    (8/14) Entropy(S(Weak)) - (6/14)
    Entropy(S(Strong))
- = 0.94 - (8/14)0.811 - (6/14)1.00 = 0.048

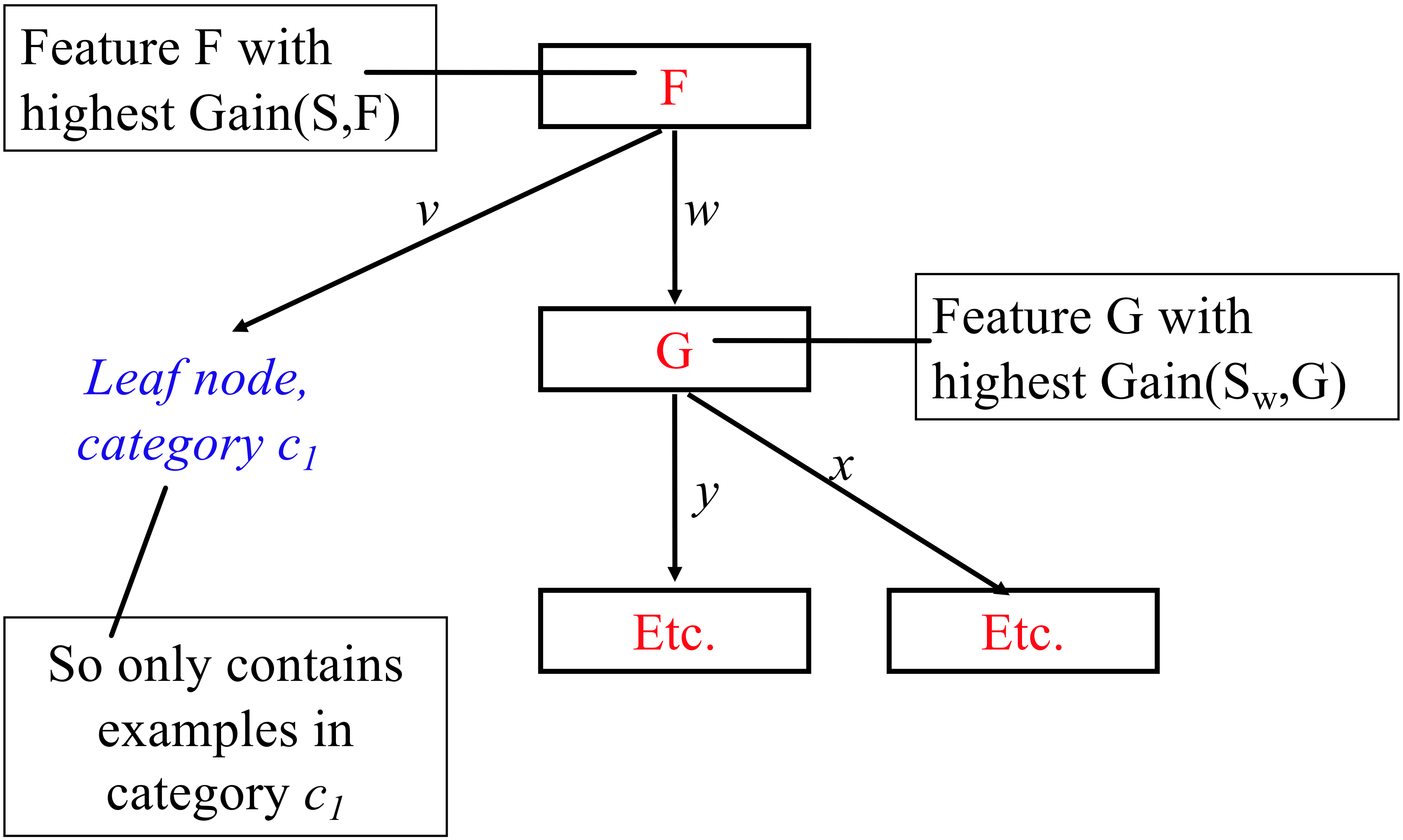
$$\mathrm{Gain}(S,F) = \mathrm{Entropy}(S) - \sum_{f \in values(F)} \frac{|S_f|}{|S|} \mathrm{Entropy}(S_f)$$

Day	Outlook	Temp	Humid	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

ID3 (Quinlan)

- Search over possible trees
  - Greedy search - no backtracking
  - Susceptible to local minima
  - Uses all features - no pruning

ID3


Feature F with highest Gain(S,F)
F
v
w
Leaf node, category c_1
G
Feature G with highest Gain(S_w,G)
y
x
So only contains examples in category c_1
Etc.
Etc.

Inductive Bias

- How does the algorithm generalize from the training examples?
  - Choose features with highest information gain
  - Minimize amount of information is left
  - Bias towards shorter trees
  - Occam's Razor
  - Put most useful features near root

Occam Razor (Simple tree)

How to come up with simple trees?

Complexity	Train error	Validation error
Simple	0.23	0.24
Moderate	0.12	0.15
Complex	0.07	0.15
Super complex	0	0.18

C4.5

- Improved version of ID3, also by Quinlan
- Use a validation set to avoid overfitting
  - Could just stop choosing features (early stopping)
- Better results from post-pruning
  - Make whole tree
  - Chop off some parts of tree afterwards

Occam Razor (Simple tree)

How to come up with simple trees?

1) Early stop
2) Pruning

Complexity	Train error	Validation error
Simple	0.23	0.24
Moderate	0.12	0.15
Complex	0.07	0.15
Super complex	0	0.18

Post-Pruning

- Run over tree
- Prune each node by replacing subtree below with a leaf
- Evaluate error (on validation data) and keep if error same or better

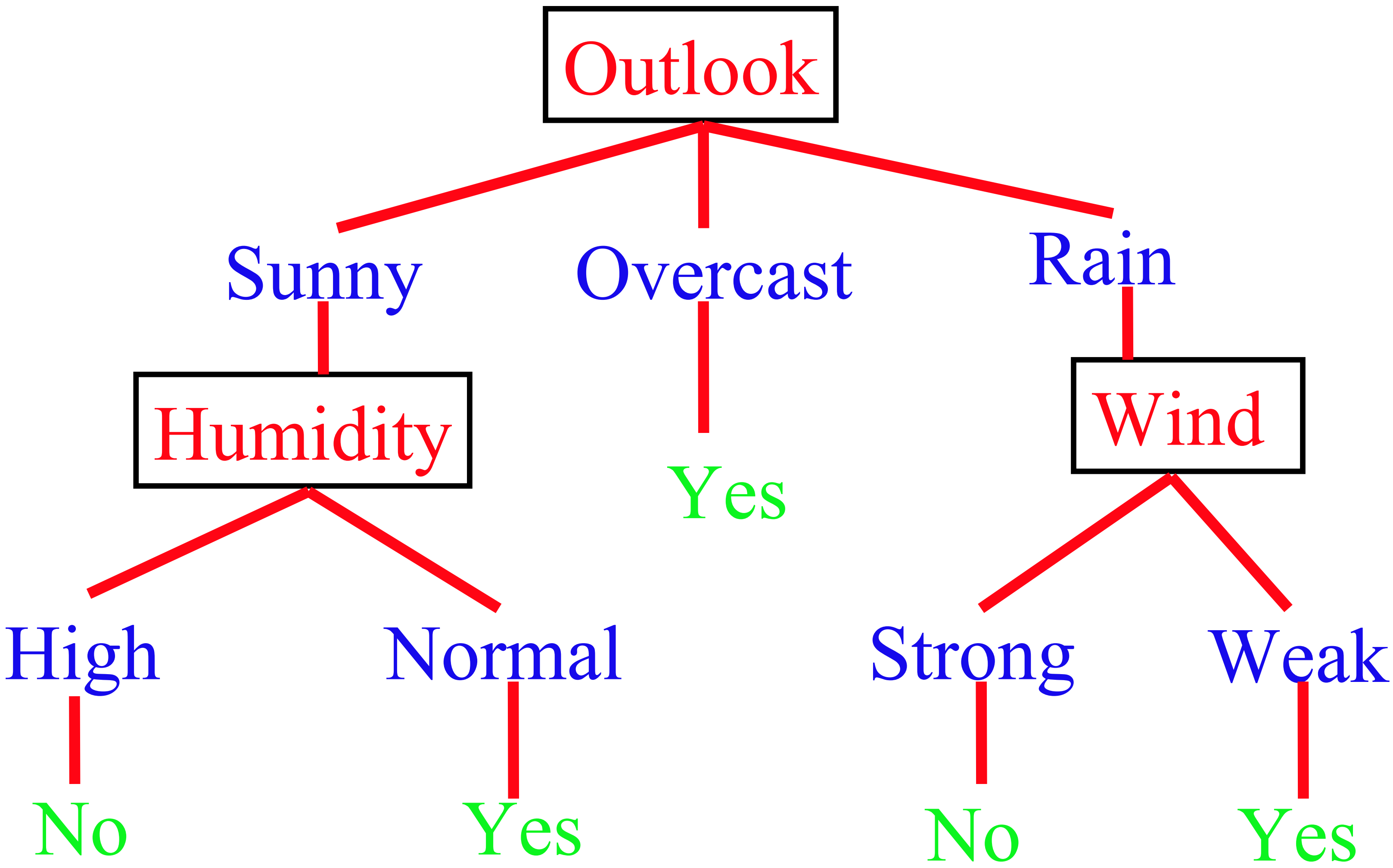
Rule Post-Pruning

- Turn tree into set of if-then rules
- Remove preconditions from each rule in turn, and check accuracy
- Sort rules according to accuracy
- Rules are easy to read

Rule Post-Pruning

- IF ((outlook = sunny) & (humidity = high))
- THEN playTennis = no
- Remove preconditions:
  - Consider IF (outlook = sunny)
  - And IF (humidity = high)
  - Test accuracy
  - If one of them is better, try removing both

ID3 Decision Tree



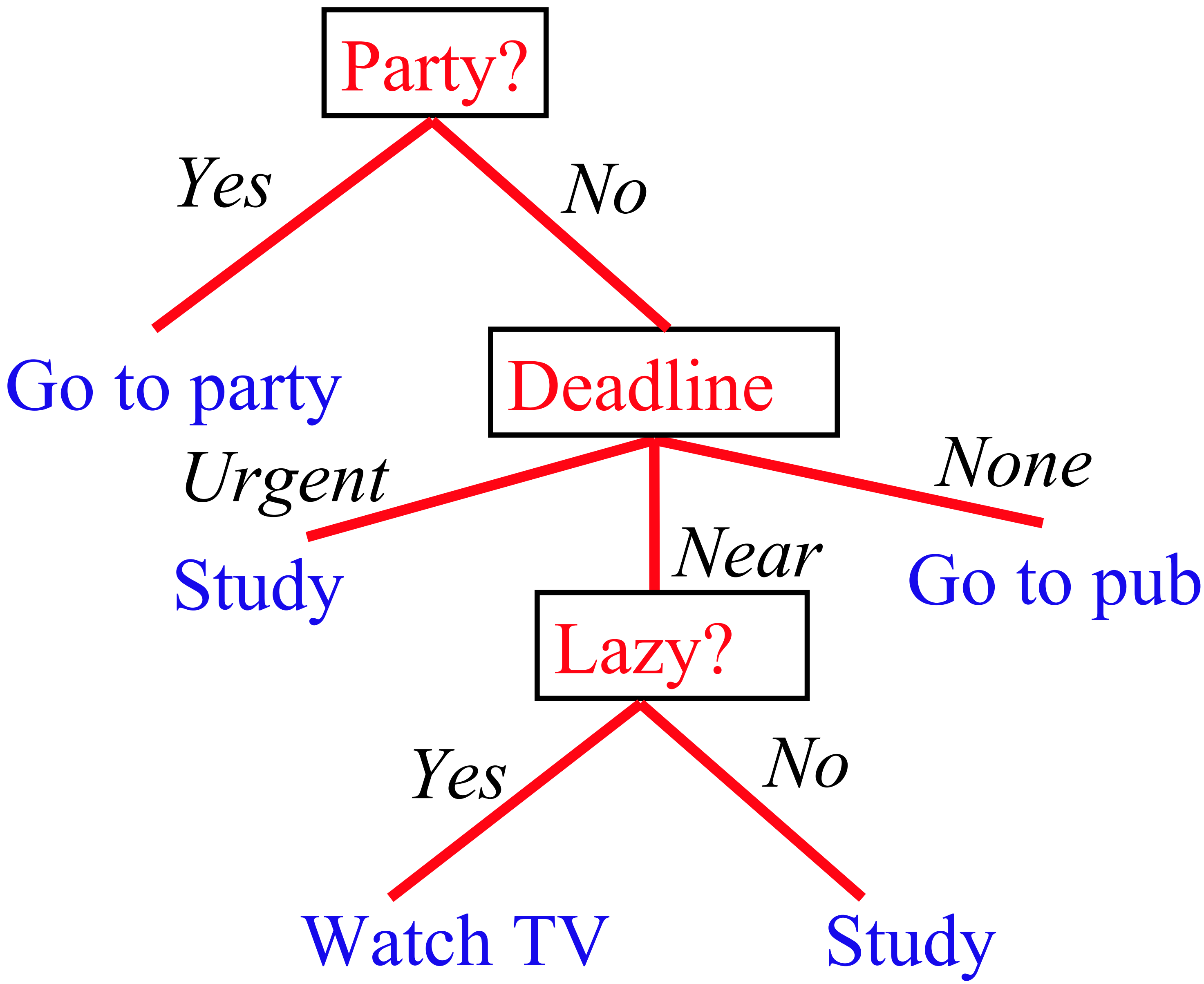
Test Case

- Outlook = Sunny
- Temperature = Cool
- Humidity = High
- Wind = Strong

Party Example,

Construct a decision tree based on these data:

Deadline,	Party,	Lazy,	Activity
Urgent,	Yes,	Yes,	Party
Urgent,	No,	Yes,	Study
Near,	Yes,	Yes,	Party
None,	Yes,	No,	Party
None,	No,	Yes,	Pub
None,	Yes,	No,	Party
Near,	No,	No,	Study
Near,	No,	Yes,	TV
Near,	Yes,	Yes,	Party
Urgent,	No,	No,	Study


Party?
Yes
No
Go to party
Deadline
Urgent
None
Near
Study
Go to pub
Lazy?
Yes
No
Watch TV
Study

Measure of Impurity: GINI

Gini Index for a given node t :

$$\boxed{GINI(t) = 1 - \sum_j [p(j \mid t)]^2}$$

$$\sum_{i=1}^{J} p_i(1-p_i) = \sum_{i=1}^{J}(p_i - {p_i}^2) = \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} {p_i}^2 = 1 - \sum_{i=1}^{J} {p_i}^2$$

(NOTE: *p(j | t)* is the relative frequency of class j at node t).

Maximum ($1 - 1/n_c$) when records are equally distributed among all classes, implying least interesting information

Minimum (0.0) when all records belong to one class, implying most interesting information

C1	**0**
C2	**6**
Gini=0.000	

C1	**1**
C2	**5**
Gini=0.278	

C1	**2**
C2	**4**
Gini=0.444	

C1	**3**
C2	**3**
Gini=0.500	

Examples for computing GINI

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

C1	**0**
C2	**6**

$P(C1) = 0/6 = 0$ $\quad$ $P(C2) = 6/6 = 1$

$Gini = 1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0$

C1	**1**
C2	**5**

$P(C1) = 1/6$ $\quad$ $P(C2) = 5/6$

$Gini = 1 - (1/6)^2 - (5/6)^2 = 0.278$

C1	**2**
C2	**4**

$P(C1) = 2/6$ $\quad$ $P(C2) = 4/6$

$Gini = 1 - (2/6)^2 - (4/6)^2 = 0.444$

Splitting Based on GINI

Used in CART, SLIQ, SPRINT.

When a node p is split into k partitions (children), the quality of split is computed as,

$$\boxed{GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)}$$

where, n_i = number of records at child i,

n = number of records at node p.

Fixes the problem that the entropy gain really likes a large number of small classes

Amazing visualization

http://www.r2d3.us/visual-intro-to-machine-learning-part-1/

Regression Trees

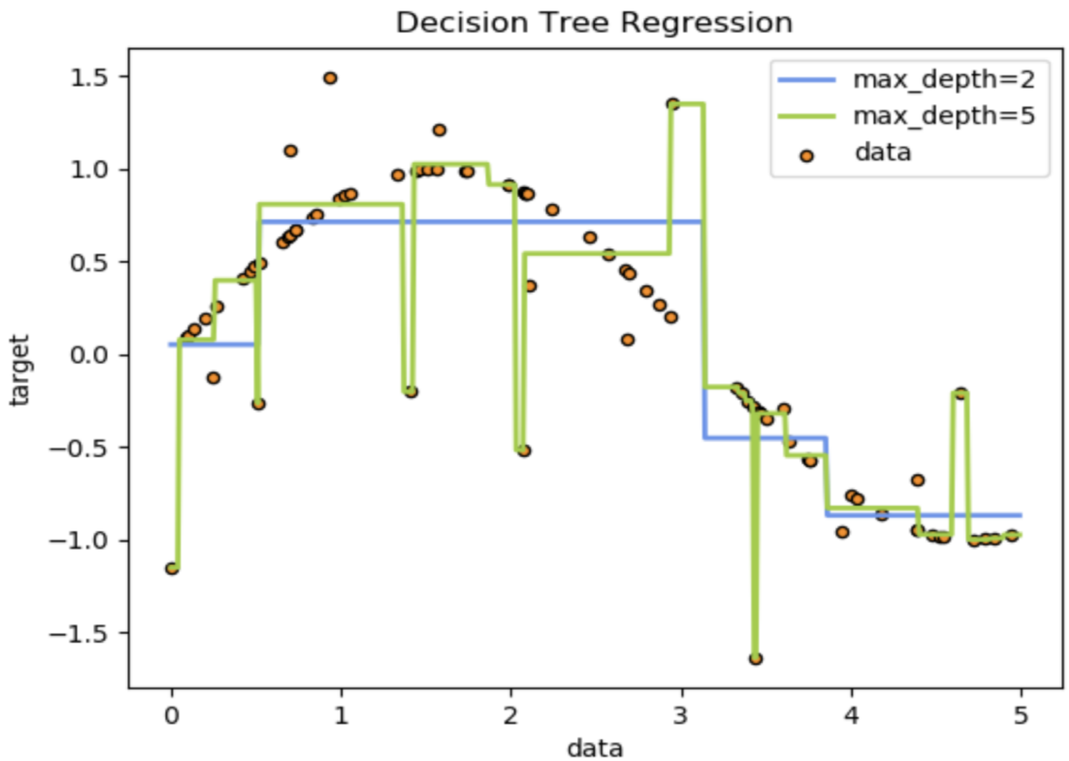
CART: Classification and Regression Trees

Assume labels are continuous: $y_i \in \mathbb{R}$
<u>Impurity: Squared Loss</u>

$$L(S) = \frac{1}{|S|} \sum_{(x,y)\in S} (y - \bar{y}_S)^2 \leftarrow \text{Average squared difference from average label}$$

$$\text{where } \bar{y}_S = \frac{1}{|S|} \sum_{(x,y)\in S} y \leftarrow \text{Average label}$$

At leaves, predict $\bar{y}_S$. Finding best split only costs $O(nlogn)$



Demo

https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/