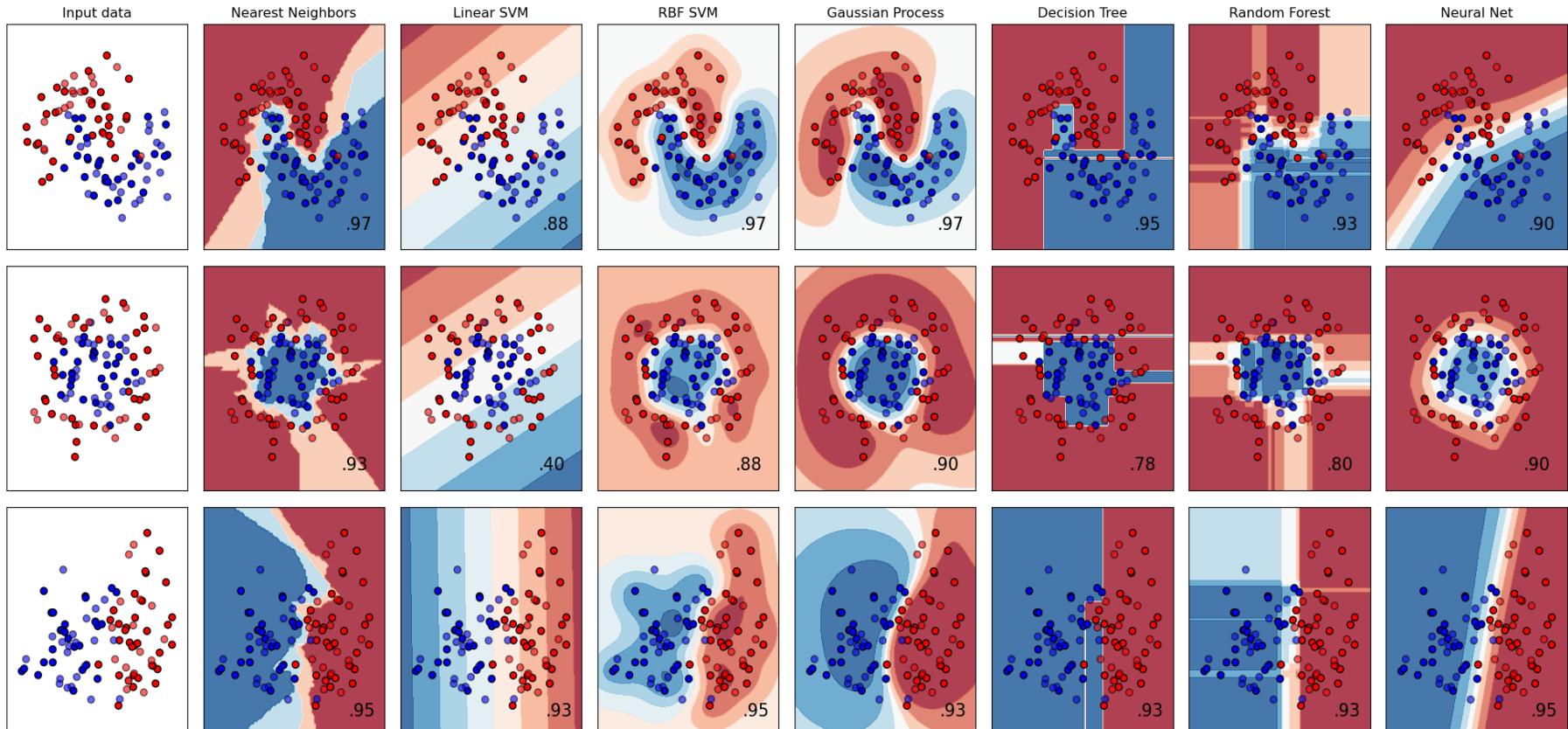


Chapter 11

Decision Tree



Tree-based Models



Induction of Decision Trees

J.R. QUINLAN

(munnari!nswitgould.oz!quinlan@seismo.css.gov)

*Centre for Advanced Computing Sciences, New South Wales Institute of Technology, Sydney 2007,
Australia*

(Received August 1, 1985)

Key words: classification, induction, decision trees, information theory, knowledge acquisition, expert systems

Abstract. The technology for building knowledge-based systems by inductive inference from examples has been demonstrated successfully in several practical applications. This paper summarizes an approach to synthesizing decision trees that has been used in a variety of systems, and it describes one such system, ID3, in detail. Results from recent studies show ways in which the methodology can be modified to deal with information that is noisy and/or incomplete. A reported shortcoming of the basic algorithm is discussed and two means of overcoming it are compared. The paper concludes with illustrations of current research directions.

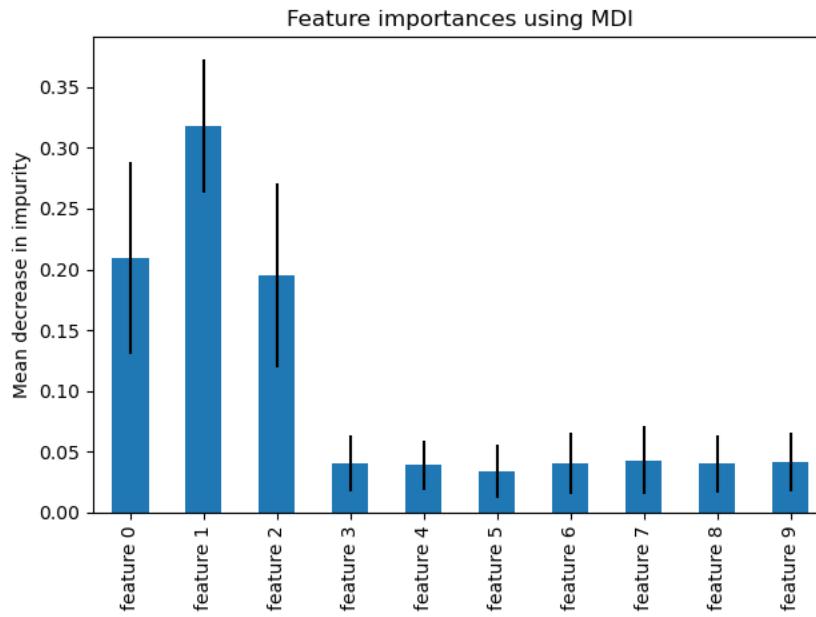


Tree-based Models

Feature importances with a forest of trees

This example shows the use of a forest of trees to evaluate the importance of features on an artificial classification task. The blue bars are the feature importances of the forest, along with their inter-trees variability represented by the error bars.

As expected, the plot suggests that 3 features are informative, while the remaining are not.



impurity-based importance
accumulation of the impurity
decrease within each tree
(random forest)

Outline

- Basic Process
- Split Selection
- Pruning
- Continuous and Missing Values
- Multivariate Decision Trees



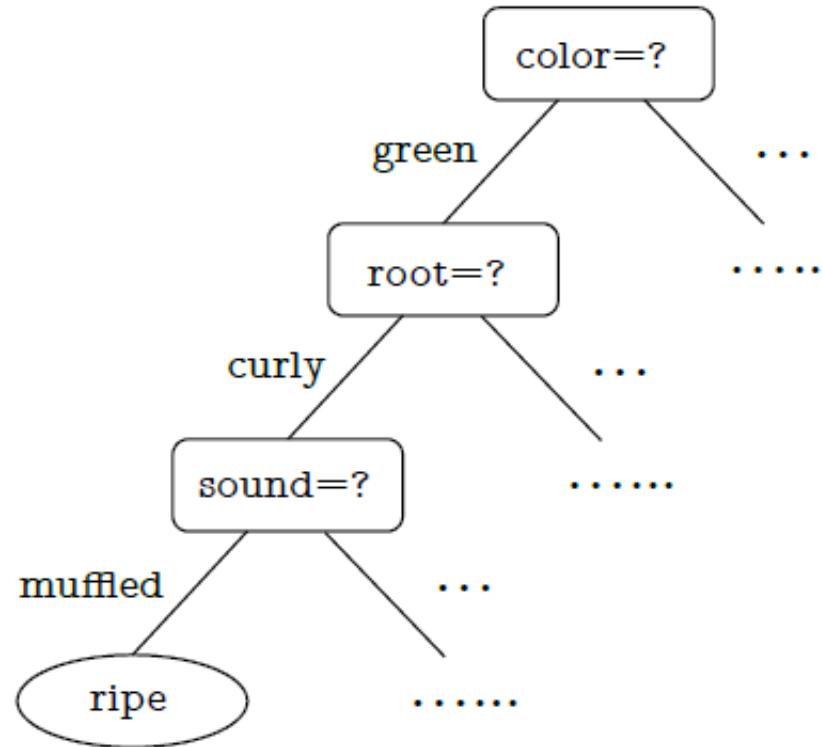
Characteristics of Decision Trees

- Non-linear classifier / regression.
- Easy to use.
- Easy to interpret.
- Susceptible to overfitting but can be avoided.



Basic Process

- A decision tree makes decisions based on tree structures.



Basic Process

- Every question asked in the decision process is a “test” on one feature.
- The conclusions at the end of the decision process correspond to the possible classifications.
- Every test leads to either the conclusion or an additional test conditioned on the current answer.
- Each path from the root node to the leaf node is a decision sequence.

The goal is to produce a tree that can
generalize to predict unseen samples.

Basic Process

Algorithm 4.1 Decision Tree Learning.

Input: Training set $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$;
Feature set $A = \{a_1, a_2, \dots, a_d\}$.

Process: Function TreeGenerate(D, A)

```
1: Generate node  $i$ ;  
2: if All samples in  $D$  belong to the same class  $C$  then  
3:   Mark node  $i$  as a class  $C$  leaf node; return  
4: end if  
5: if  $A = \emptyset$  OR all samples in  $D$  take the same value on  $A$  then  
6:   Mark node  $i$  as a leaf node, and its class label is the majority class in  $D$ ; return  
7: end if  
8: Select the optimal splitting feature  $a_*$  from  $A$ ;  
9: for each value  $a_*^v$  in  $a_*$  do  
10:   Generate a branch for node  $i$ ; Let  $D_v$  be the subset of samples taking value  $a_*^v$  on  $a_*$ ;  
11:   if  $D_v$  is empty then  
12:     Mark this child node as a leaf node, and label it with the majority class in  $D$ ; return  
13:   else  
14:     Use TreeGenerate( $D_v, A \setminus \{a_*\}$ ) as the child node.  
15:   end if  
16: end for
```

Output: A decision tree with root node i .

(1) All samples in the current node belong to the same class.

(2) The current feature set is empty, or all samples have the same feature values.

(3) There is no sample in the current node.



Outline

- Basic Process
- Split Selection
- Pruning
- Continuous and Missing Values
- Multivariate Decision Trees



Split Selection

- The core of the decision tree learning algorithm is **selecting the optimal splitting feature**. Generally speaking, as the splitting process proceeds, we wish more samples within each node to **belong to a single class**, that is, **increasing the purity** of each node.
- Classical Guidelines for Selecting the Splitting Features:
 - Information Gain
 - Gain Ratio
 - Gini Index



Split Selection: Information Gain

- One of the most commonly used measures for purity is information entropy, or simply **entropy**. Let p_k denote the proportion of the k -th class in the current data set D , where $k = 1, 2, \dots, |\mathcal{Y}|$. Then, the entropy is defined as

$$\text{Ent}(D) = - \sum_{k=1}^{|\mathcal{Y}|} p_k \log_2 p_k$$

The lower the $\text{Ent}(D)$, the higher the purity of D .

- In the calculation of entropy, $p \log_2 p = 0$ when $p = 0$.
- The minimum of $\text{Ent}(D)$ is 0 and the maximum is $\log_2 |\mathcal{Y}|$.

Split Selection: Information Gain

- Suppose that the discrete feature a has V possible values $\{a^1, a^2, \dots, a^V\}$. Then, splitting the data set D by feature a produces V child nodes, where the v th child node D^v includes all samples in D taking the value a^v for feature a . Then, the *information gain* of splitting the data set D with feature a is calculated as

$$\text{Gain}(D, a) = \text{Ent}(D) - \sum_{v=1}^V \frac{|D^v|}{|D|} \text{Ent}(D^v)$$

is the importance of each node. The greater the number of samples, the greater the impact of the branch node.

- In general, **the higher the information gain, the more purity improvement** we can expect by splitting D with feature a .
- The decision tree algorithm ID₃ [Quinlan, 1986] takes information gain as the guideline for selecting the splitting features.

What is Information?

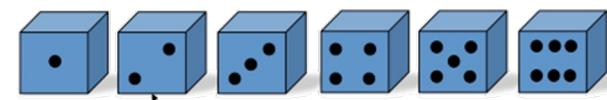
- If the probability of this event happening is small and it happens, the information is large.
- Observing the outcome of a coin flip is head.

$$I = -\log_2 \frac{1}{2} = 1$$



- Observe the outcome of a dice is 6.

$$I = -\log_2 \frac{1}{6} = 2.58$$



What is Entropy?

- The expected amount of information when observing the output of a random variable X .

$$H(X) = E(I(X)) = \sum_i p(x_i)I(x_i) = -\sum_i p(x_i)\log_2 p(x_i)$$

- If the X can have 8 outcomes and all are equally likely.

$$H(X) = -\sum_i \frac{1}{8} \log_2 \frac{1}{8} = 3$$

Split Selection: Information Gain

■ A Concrete Example

ID	color	root	sound	texture	umbilicus	surface	ripe
1	green	curly	muffled	clear	hollow	hard	true
2	dark	curly	dull	clear	hollow	hard	true
3	dark	curly	muffled	clear	hollow	hard	true
4	green	curly	dull	clear	hollow	hard	true
5	light	curly	muffled	clear	hollow	hard	true
6	green	slightly curly	muffled	clear	slightly hollow	soft	true
7	dark	slightly curly	muffled	slightly blurry	slightly hollow	soft	true
8	dark	slightly curly	muffled	clear	slightly hollow	hard	true
9	dark	slightly curly	dull	slightly blurry	slightly hollow	hard	false
10	green	straight	crisp	clear	flat	soft	false
11	light	straight	crisp	blurry	flat	hard	false
12	light	curly	muffled	blurry	flat	soft	false
13	green	slightly curly	muffled	slightly blurry	hollow	hard	false
14	light	slightly curly	dull	slightly blurry	hollow	hard	false
15	dark	slightly curly	muffled	clear	slightly hollow	soft	false
16	light	curly	muffled	blurry	flat	hard	false
17	green	curly	dull	slightly blurry	slightly hollow	hard	false

This data set includes 17 training samples. 8/17 of them are positive and 9/17 of them are negative. the entropy of the root node is

$$\text{Ent}(D) = - \sum_{k=1}^2 p_k \log_2 p_k = -\left(\frac{8}{17} \log_2 \frac{8}{17} + \frac{9}{17} \log_2 \frac{9}{17}\right) = 0.998$$

Split Selection: Information Gain

- If D is split by color, then there are 3 subsets: D^1 (color = green), D^2 (color = dark) and D^3 (color = light).
- Subset D^1 includes 6 samples $\{1, 4, 6, 10, 13, 17\}$, in which $p_1 = \frac{3}{6}$ of them are positive and $p_2 = \frac{3}{6}$ of them are negative. D^2 and D^3 can be discussed similarly. The entropy of the 3 child nodes are

$$\text{Ent}(D^1) = -\left(\frac{3}{6} \log_2 \frac{3}{6} + \frac{3}{6} \log_2 \frac{3}{6}\right) = 1.000$$

$$\text{Ent}(D^2) = -\left(\frac{4}{6} \log_2 \frac{4}{6} + \frac{2}{6} \log_2 \frac{2}{6}\right) = 0.918$$

$$\text{Ent}(D^3) = -\left(\frac{1}{5} \log_2 \frac{1}{5} + \frac{4}{5} \log_2 \frac{4}{5}\right) = 0.722$$

- The information gain of splitting by color is

$$\begin{aligned}\text{Gain}(D, \text{color}) &= \text{Ent}(D) - \sum_{v=1}^3 \frac{|D^v|}{|D|} \text{Ent}(D^v) \\ &= 0.998 - \left(\frac{6}{17} \times 1.000 + \frac{6}{17} \times 0.918 + \frac{5}{17} \times 0.722\right) \\ &= 0.109\end{aligned}$$

Split Selection: Information Gain

- Similarly, we calculate the information gain of other features:

$$\text{Gain}(D, \text{root}) = 0.143;$$

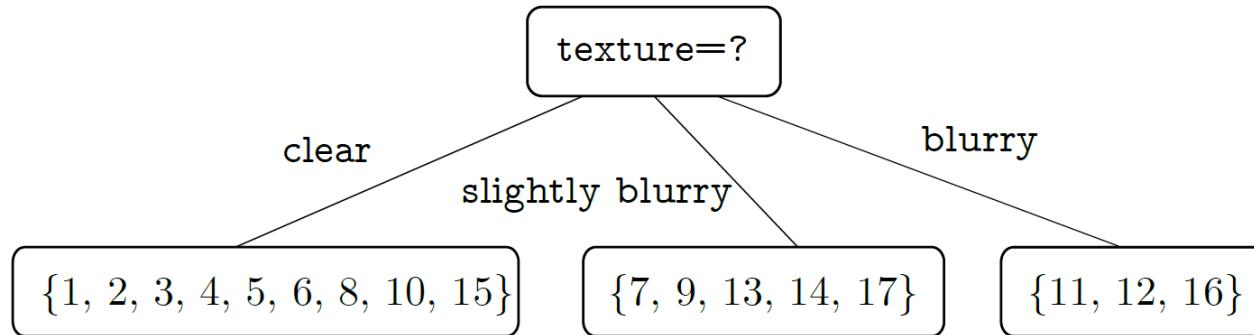
$$\text{Gain}(D, \text{texture}) = 0.381;$$

$$\text{Gain}(D, \text{surface}) = 0.006.$$

$$\text{Gain}(D, \text{sound}) = 0.141;$$

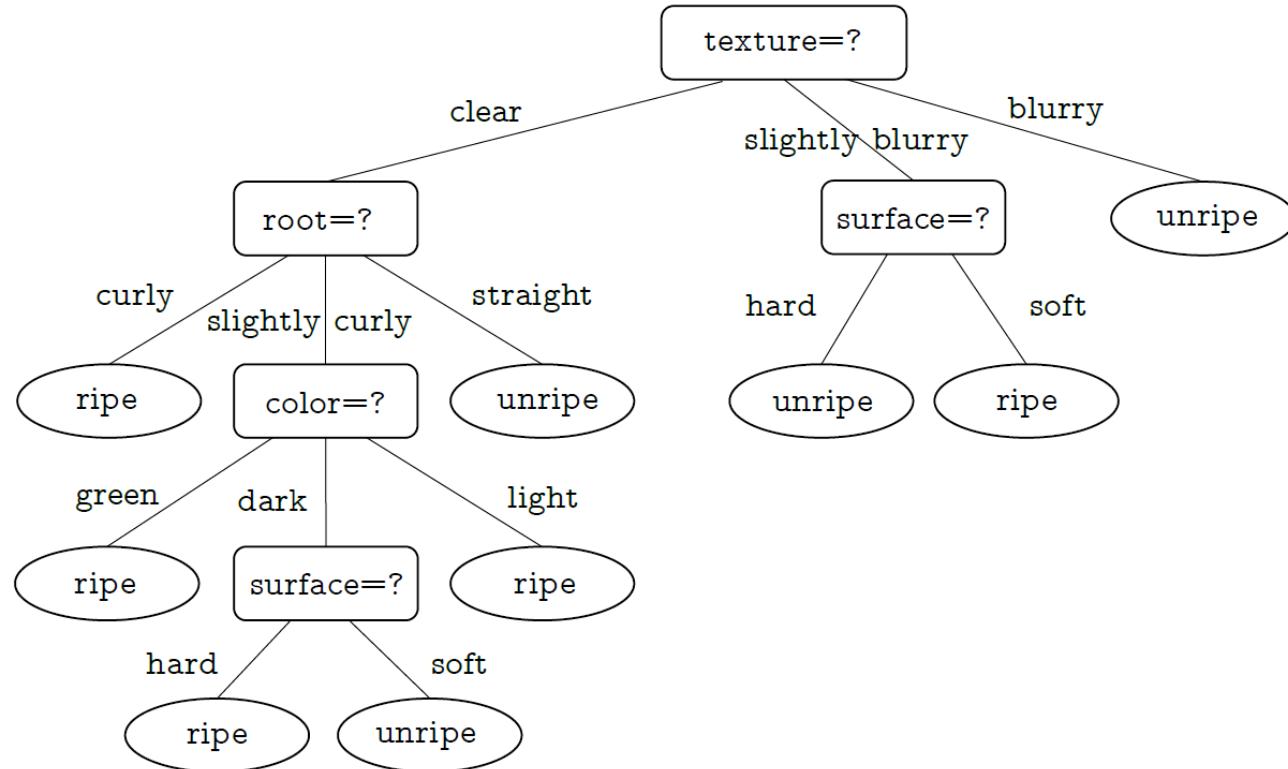
$$\text{Gain}(D, \text{umbilicus}) = 0.289;$$

- Since splitting by **texture** gives the highest information gain, it is chosen as the splitting feature.



Split Selection: Information Gain

- Then, each child node is further split by the decision tree algorithm. We can obtain the final decision tree:



ID₃ Algorithm

Algorithm 1 ID₃

```
procedure ID3( $T$ )
    if  $T$  only consists of the same target values then
        return Leaf with target value
    else if  $T$  has the same values for the same attributes but
        different target values then
        return Leaf with target value that occurs most
    end if
     $a \leftarrow$  Attribute in  $T$  with highest information gain
     $node \leftarrow$  Node with attribute  $a$ 
     $partitions \leftarrow$  PARTITION( $T, a$ )
    for  $partition$  in  $partitions$  do
         $subtree \leftarrow ID_3(partition)$ 
        Add  $subtree$  to  $node$ 
    end for
    return  $node$ 
end procedure
```

ID₃ Algorithm

Algorithm 2 Partitioning

```
procedure PARTITION( $T, a$ )
    partitions  $\leftarrow \emptyset$ 
    for every value  $v$  that the attribute  $a$  can have in  $T$  do
        partition  $\leftarrow$  all sets in  $T$  that have the value  $v$  on
        attribute  $a$ 
        Add partition to partitions
    end for
    return partitions
end procedure
```

Split Selection: Information Gain

Bias

- If we consider ID as a candidate splitting feature, its information gain will be much higher than that of any other features in general. However, such a decision tree does not have generalization ability and cannot effectively predict new samples.

It turns out that the information gain criterion is biased towards features with more possible values.



Split Selection: Gain Ratio

- The *gain ratio (normalized)* of feature a is defined as

$$\text{Gain_ratio}(D, a) = \frac{\text{Gain}(D, a)}{\text{IV}(a)}$$

where

$$\text{IV}(a) = - \sum_{v=1}^V \frac{|D^v|}{|D|} \log_2 \frac{|D^v|}{|D|}$$

$$\text{Gain}(D, a) = \text{Ent}(D) - \sum_{v=1}^V \frac{|D^v|}{|D|} \text{Ent}(D^v)$$

is called the *intrinsic information (value)* of feature a . It penalizes features with a large number of distinct values, favoring simpler splits.

Gain ratio is biased towards features with fewer possible values.

- The C4.5 algorithm [Quinlan, 1993] uses a heuristic method : selecting the feature with the highest gain ratio from the set of candidate features with an information gain above the average.



Split Selection: Gini Index

- The Gini value of data set D is defined as

$$\text{Gini}(D) = \sum_{k=1}^{|\mathcal{Y}|} \sum_{k' \neq k} p_k p_{k'} = 1 - \sum_{k=1}^{|\mathcal{Y}|} p_k^2$$

the probability of incorrectly classifying a randomly chosen sample if it is randomly labeled according to the class distribution.

The lower the $\text{Gini}(D)$, the higher the purity of data set D .

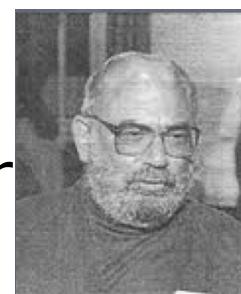
- The *Gini index (impurity)* of feature a is defined as

$$\text{Gini_index}(D, a) = \sum_{v=1}^V \frac{|D^v|}{|D|} \text{Gini}(D^v)$$

- We select the feature with the lowest Gini index as the splitting feature

$$a_* = \operatorname{argmin}_{a \in A} \text{Gini_index}(D, a)$$

- CART [Breiman et al., 1984] employs Gini index for selecting the splitting feature.



Can you provide an example of how Gini impurity is calculated?



ChatGPT

Poe

Certainly! Let's consider a simple example where we have a dataset with 10 elements and two classes: Class A and Class B. Here's the distribution of the classes within the dataset:

Class A: 6 elements

Class B: 4 elements

To calculate the Gini impurity, we follow these steps:

1. Calculate the probability of each class label:

- Probability of Class A: $6/10 = 0.6$
- Probability of Class B: $4/10 = 0.4$

2. Calculate the Gini impurity using the formula:

$$\text{Gini impurity} = 1 - (p_1^2 + p_2^2)$$

$$\text{Gini impurity} = 1 - (0.6^2 + 0.4^2)$$

$$\text{Gini impurity} = 1 - (0.36 + 0.16)$$

$$\text{Gini impurity} = 1 - 0.52$$

$$\text{Gini impurity} = 0.48$$

So, in this example, the Gini impurity of the dataset is 0.48. This value indicates the impurity or disorder within the dataset, with a higher value representing a less pure distribution of classes.

Note that the Gini impurity can also be calculated at each internal node of a decision tree to determine the impurity of the subsets created by different splits. The split with the lowest Gini impurity is typically chosen as it leads to more homogeneous child nodes and better separation of classes.

Regression tree?

Hint:

- standard deviation is used as the metric
- leaf value typically set to the mean value

SU

information gain in regression tree



In a regression tree, the goal is to minimize the variance of the target variable within each leaf node. The information gain for a split in a regression tree is given by:

$$IG = \text{Var}(\text{parent}) - \frac{N_{\text{left}}}{N} \text{Var}(\text{left}) - \frac{N_{\text{right}}}{N} \text{Var}(\text{right})$$

where $\text{Var}(\text{parent})$ is the variance of the target variable in the parent node before the split, N is the total number of observations in the parent node, N_{left} and N_{right} are the number of observations in the left and right child nodes resulting from the split, and $\text{Var}(\text{left})$ and $\text{Var}(\text{right})$ are the variances of the target variable in the left and right child nodes, respectively.

This formula calculates the reduction in variance between the parent node and the two child nodes resulting from the split. The split that maximizes the information gain is selected as the best split for the tree.

The information gain is a commonly used criterion for selecting the best split in decision trees, and it can also be applied in other types of trees, including regression trees.

Outline

- Basic Process
- Split Selection
- Pruning
- Continuous and Missing Values
- Multivariate Decision Trees



Pruning

- Why pruning?
 - *Pruning* is the primary strategy of decision tree learning algorithms to **deal with overfitting**.
 - If there are too many branches, then the learner may be misled by the peculiarities of the training samples and incorrectly consider them as the underlying truth.
- General Pruning Strategies
 - *pre-pruning*
 - *post-pruning*
- How to evaluate generalization ability after pruning?
 - We can use the **hold-out method** to reserve part of the data as a validation set for performance evaluation.



Pruning

■ Data Set

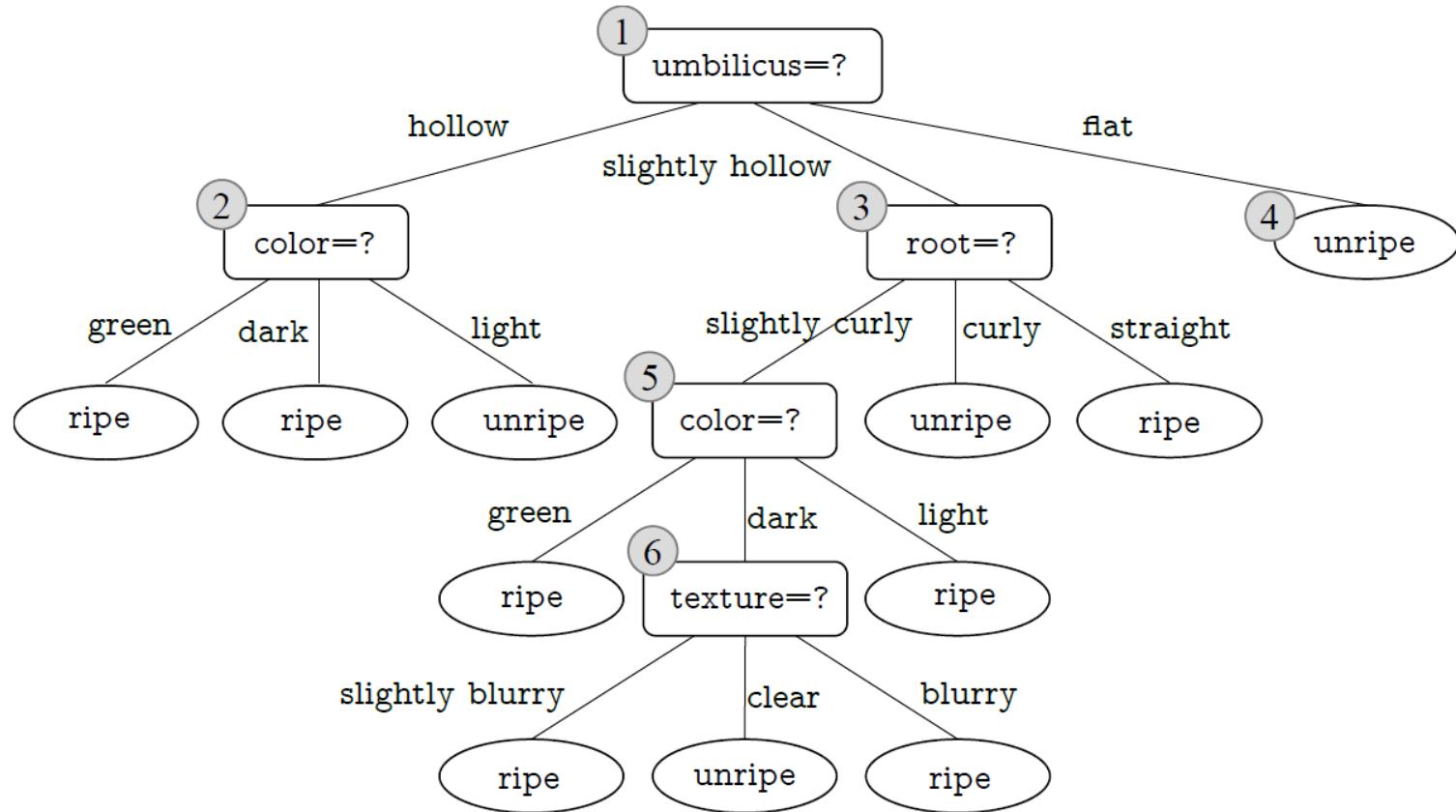
Training
Set

ID	color	root	sound	texture	umbilicus	surface	ripe
1	green	curly	muffled	clear	hollow	hard	true
2	dark	curly	dull	clear	hollow	hard	true
3	dark	curly	muffled	clear	hollow	hard	true
6	green	slightly curly	muffled	clear	slightly hollow	soft	true
7	dark	slightly curly	muffled	slightly blurry	slightly hollow	soft	true
10	green	straight	crisp	clear	flat	soft	false
14	light	slightly curly	dull	slightly blurry	hollow	hard	false
15	dark	slightly curly	muffled	clear	slightly hollow	soft	false
16	light	curly	muffled	blurry	flat	hard	false
17	green	curly	dull	slightly blurry	slightly hollow	hard	false
ID	color	root	sound	texture	umbilicus	surface	ripe
4	green	curly	dull	clear	hollow	hard	true
5	light	curly	muffled	clear	hollow	hard	true
8	dark	slightly curly	muffled	clear	slightly hollow	hard	true
9	dark	slightly curly	dull	slightly blurry	slightly hollow	hard	false
11	light	straight	crisp	blurry	flat	hard	false
12	light	curly	muffled	blurry	flat	soft	false
13	green	slightly curly	muffled	slightly blurry	hollow	hard	false

Validation
Set

Pruning

■ The Unpruned Decision Tree



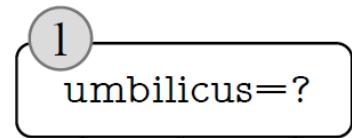
Pruning: Pre-pruning

- Pre-pruning decides by comparing **the generalization abilities before and after splitting.**
 - If the validation accuracy decreases after pruning, the splitting is accepted.
 - Otherwise, the splitting is rejected.
- When no splitting is performed, this node is marked as a leaf node and its label is set to the majority class.

Pruning: Pre-pruning

Validation Set

ID	color	root	sound	texture	umbilicus	surface	ripe
4	green	curly	dull	clear	hollow	hard	true
5	light	curly	muffled	clear	hollow	hard	true
8	dark	slightly curly	muffled	clear	slightly hollow	hard	true
9	dark	slightly curly	dull	slightly blurry	slightly hollow	hard	false
11	light	straight	crisp	blurry	flat	hard	false
12	light	curly	muffled	blurry	flat	soft	false
13	green	slightly curly	muffled	slightly blurry	hollow	hard	false



validation accuracy
“umbilicus=?” before splitting: 42.9%
after splitting: 71.4%
decision by pre-pruning: split

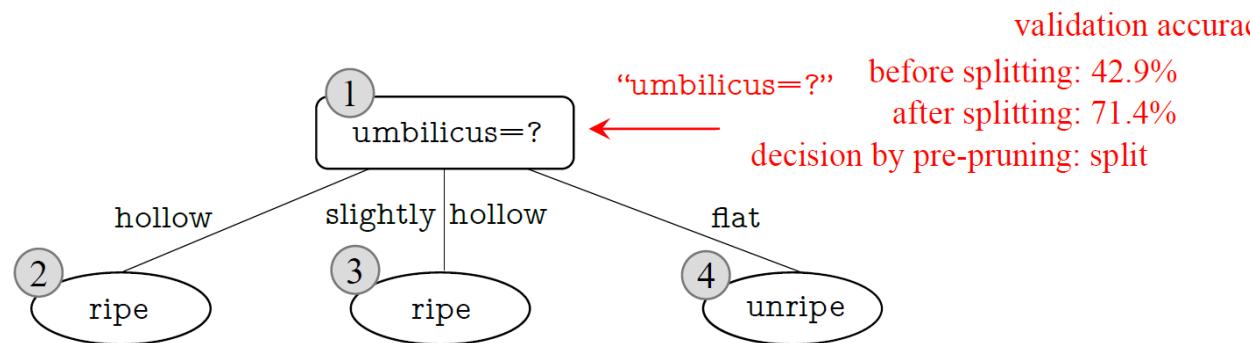
Node ① : When no splitting is performed, this node is marked as a leaf node and its label is set to the majority class (i.e., *ripe*). In validation set, {4, 5, 8} are correctly classified. Then the validation accuracy is

$$\frac{3}{7} \times 100\% = 42.9\%$$

Pruning: Pre-pruning

Validation Set

ID	color	root	sound	texture	umbilicus	surface	ripe
4	green	curly	dull	clear	hollow	hard	true
5	light	curly	muffled	clear	hollow	hard	true
8	dark	slightly curly	muffled	clear	slightly hollow	hard	true
9	dark	slightly curly	dull	slightly blurry	slightly hollow	hard	false
11	light	straight	crisp	blurry	flat	hard	false
12	light	curly	muffled	blurry	flat	soft	false
13	green	slightly curly	muffled	slightly blurry	hollow	hard	false



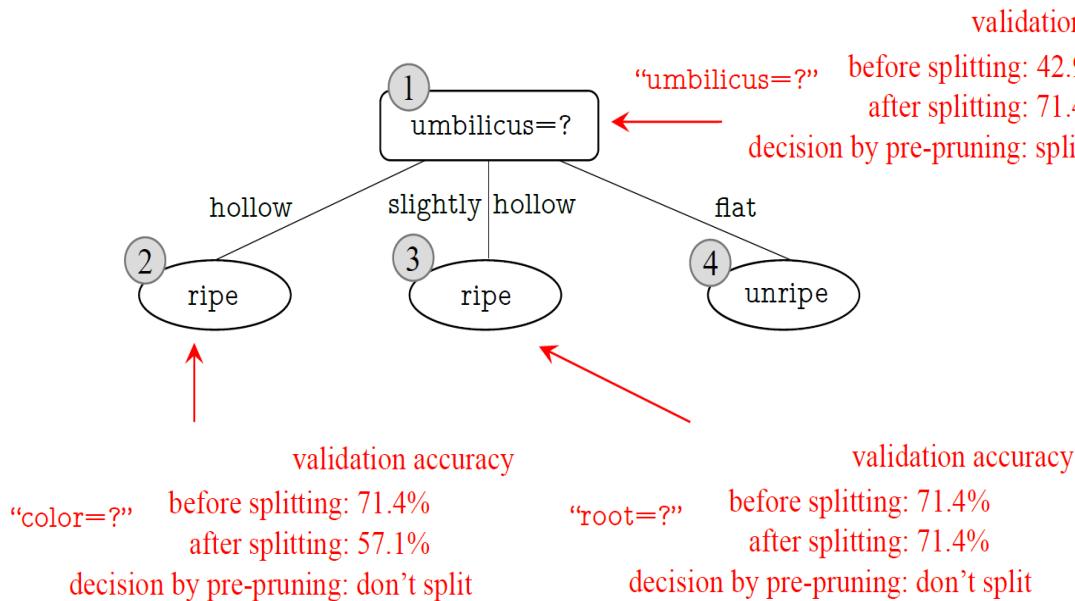
Node ① : After splitting, the samples are placed into 3 child nodes. We mark these 3 nodes as leaf nodes and set the labels to the majority classes. Then the validation accuracy is

$$\frac{5}{7} \times 100\% = 71.4\%$$

Pruning: Pre-pruning

Validation Set

ID	color	root	sound	texture	umbilicus	surface	ripe
4	green	curly	dull	clear	hollow	hard	true
5	light	curly	muffled	clear	hollow	hard	true
8	dark	slightly curly	muffled	clear	slightly hollow	hard	true
9	dark	slightly curly	dull	slightly blurry	slightly hollow	hard	false
11	light	straight	crisp	blurry	flat	hard	false
12	light	curly	muffled	blurry	flat	soft	false
13	green	slightly curly	muffled	slightly blurry	hollow	hard	false



The pre-pruning strategy stops splitting node ③ and node ② . For node ④, no splitting is needed since all samples belong to the same class. Finally, we obtain a decision tree with only one splitting. Such a decision tree is called a decision stump.

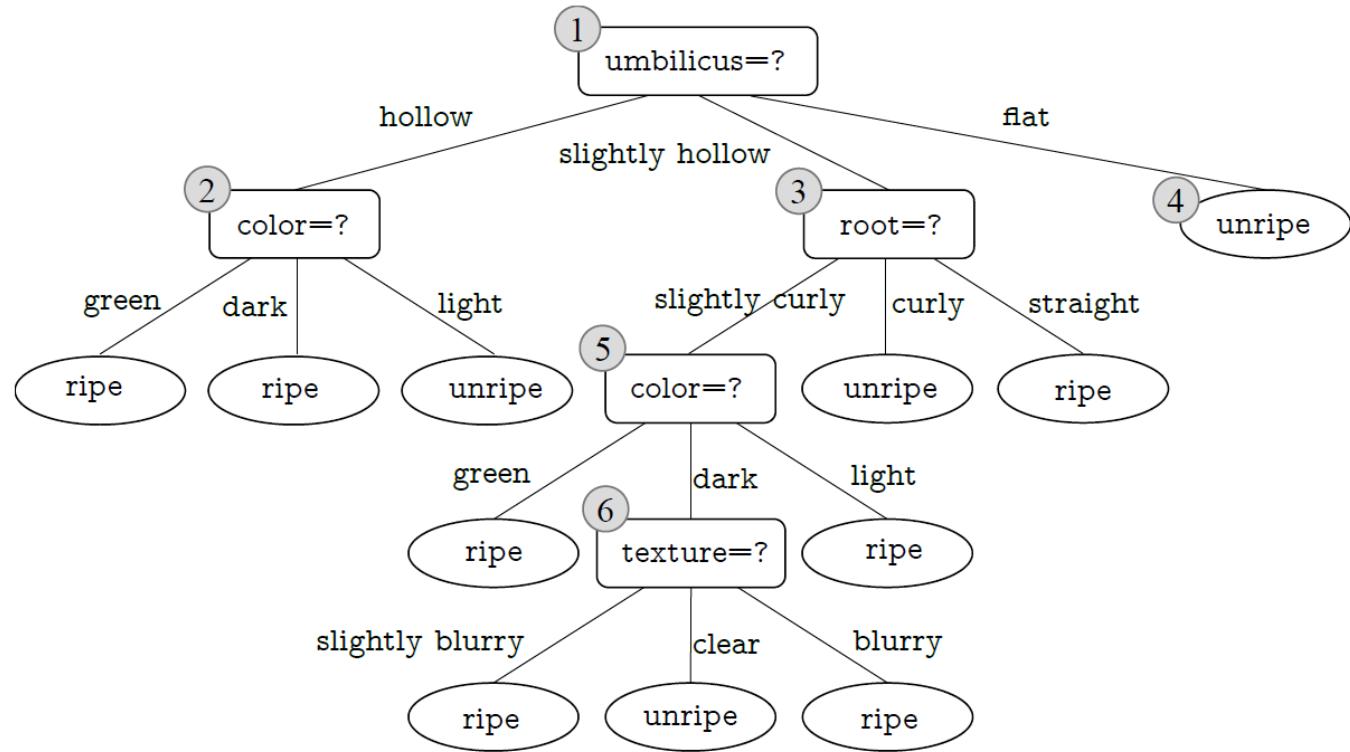
Pruning: Pre-pruning

- Advantage
 - Reduce the risk of overfitting
 - Reduce the computational cost of training and testing
- Disadvantage
 - Risk of underfitting: Although some branches are prevented by pre-pruning due to little or even negative improvement on generalization ability, it is still possible that their subsequent splits can lead to significant improvement. These branches are pruned due to the greedy nature of pre-pruning, and it may introduce the risk of underfitting.

Pruning: Post-pruning

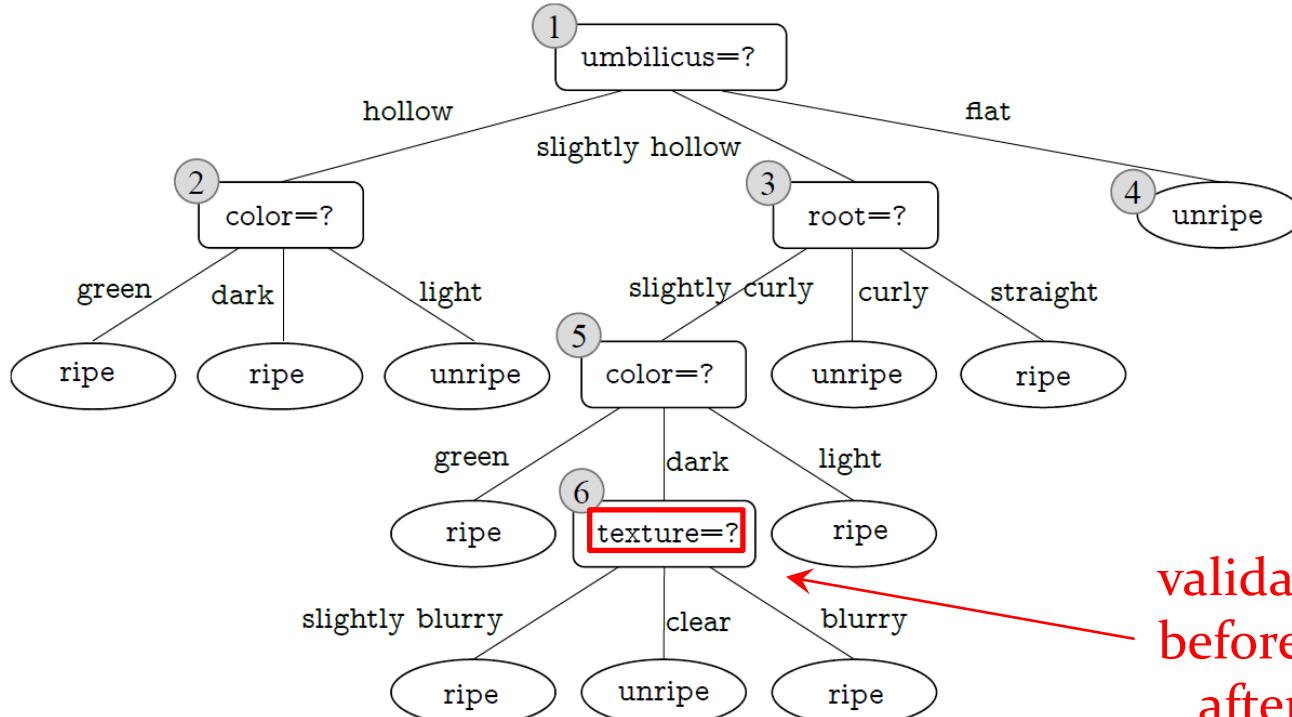
- Post-pruning allows a decision tree to grow into a complete tree. Then it takes a bottom-up strategy to examine every non-leaf node in the completely grown decision tree.

The validation accuracy of this decision tree is 42.9%



Pruning: Post-pruning

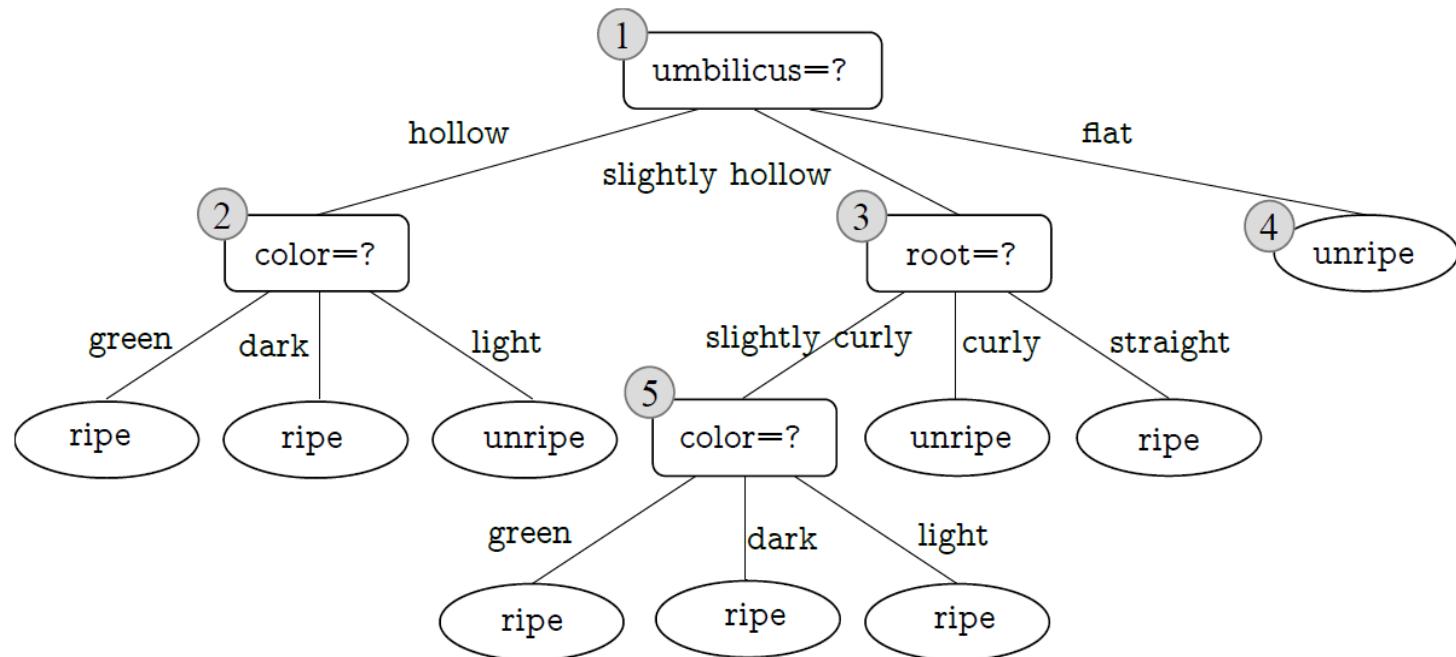
- Node ⑥ is the first one examined by post-pruning.



validation accuracy
before pruning: 42.9%
after pruning: 57.1%
decision: pruning

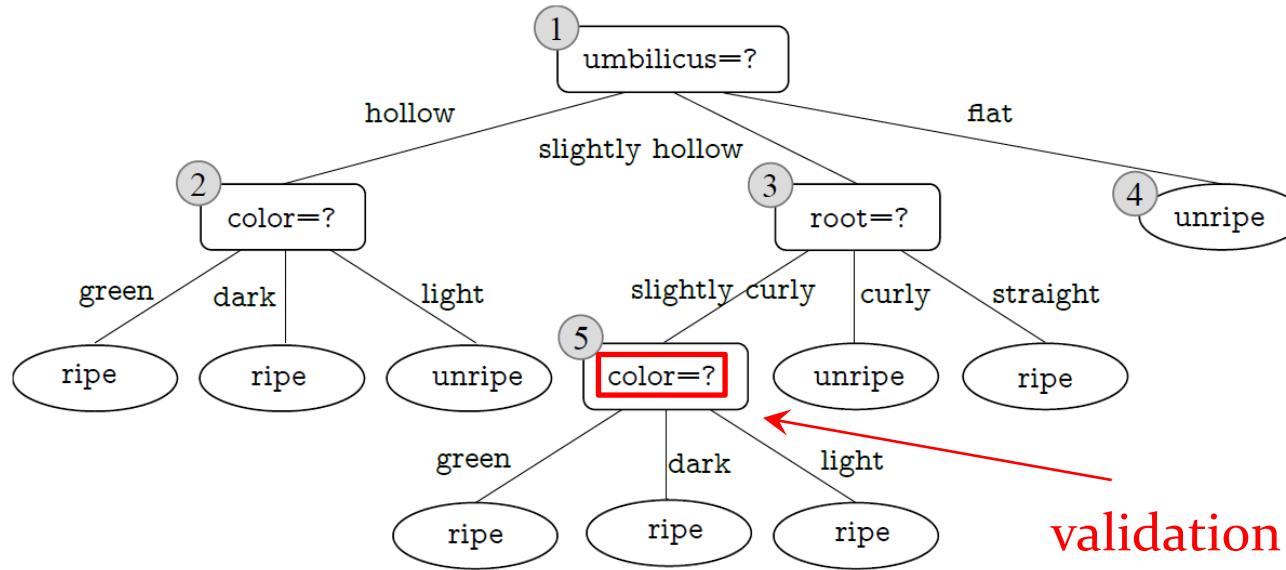
Pruning: Post-pruning

- Node ⑥ is the first one examined by post-pruning.



Pruning: Post-pruning

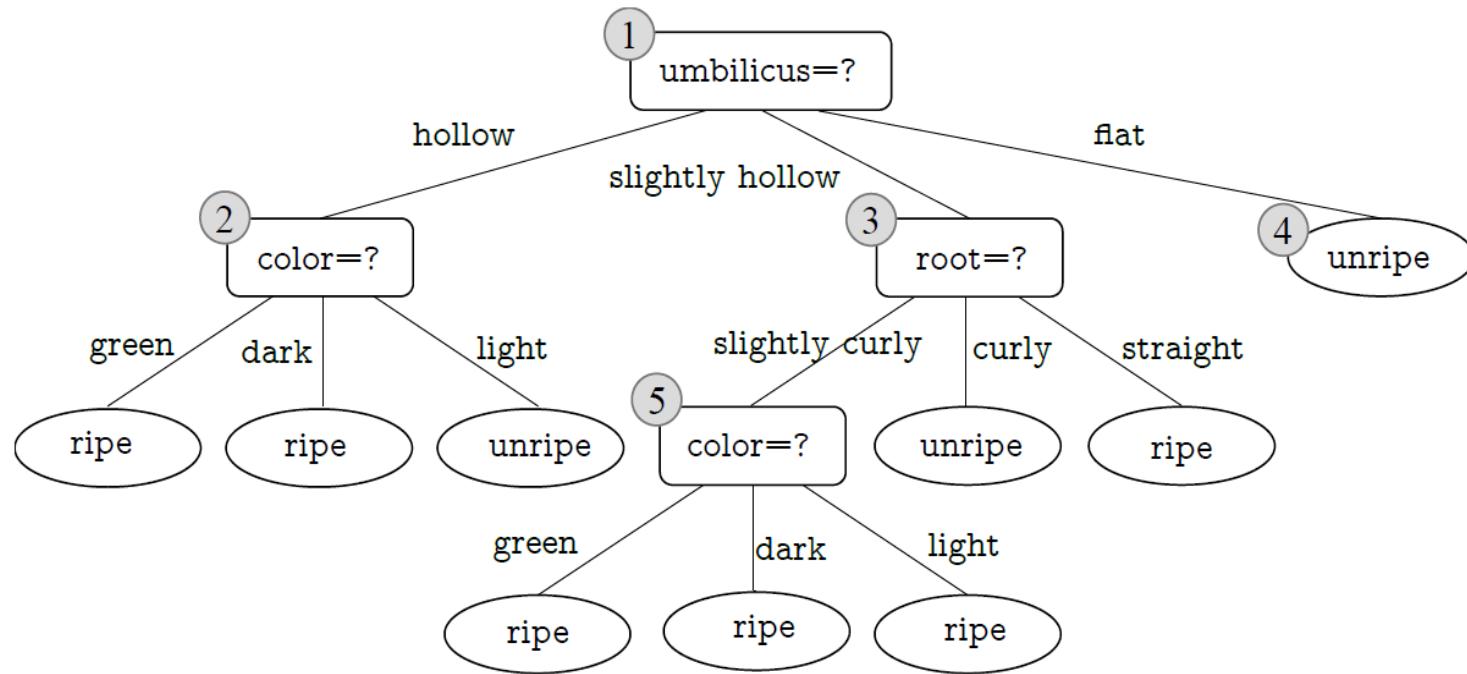
- Next, post-pruning examines node ⑤.



validation accuracy
before pruning: 57.1%
after pruning: 57.1%
decision: no pruning

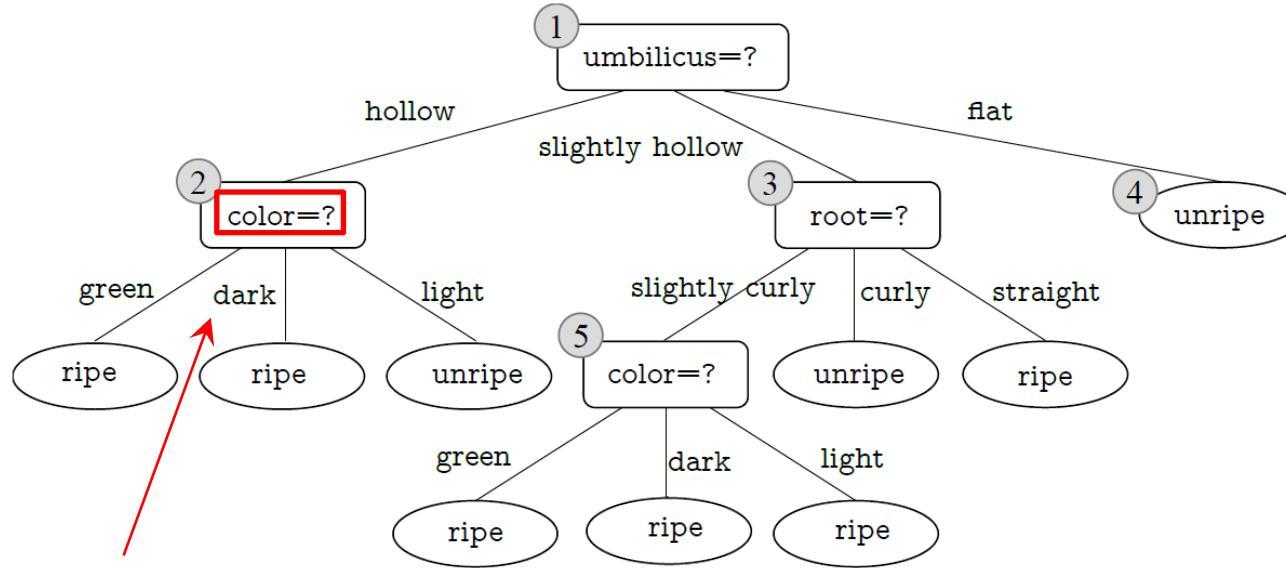
Pruning: Post-pruning

- Next, post-pruning examines node ⑤.



Pruning: Post-pruning

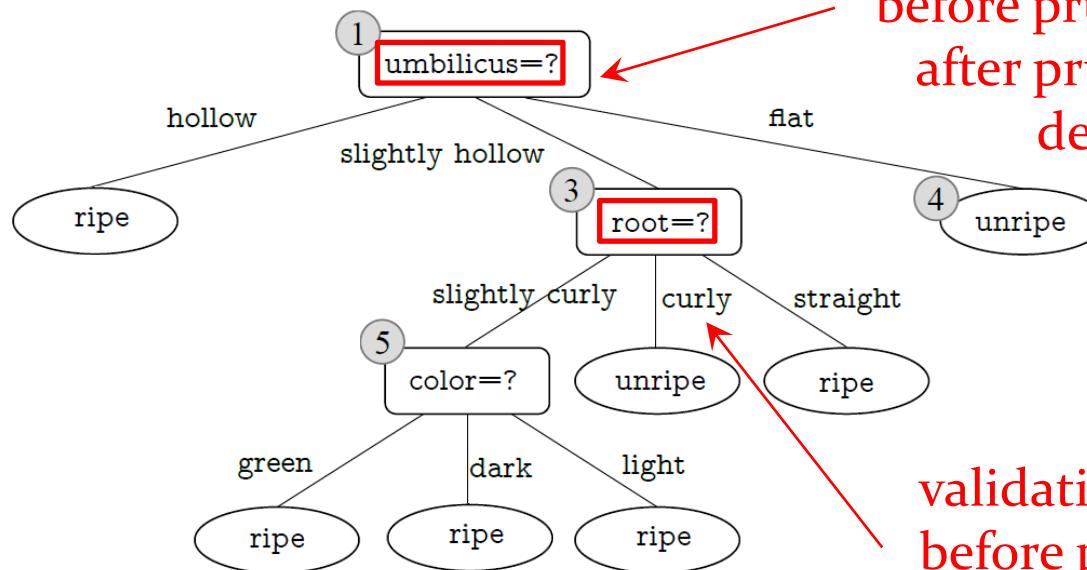
- Then, post-pruning examines node ②.



validation accuracy
before pruning: 57.1%
after pruning: 71.4%
decision: pruning

Pruning: Post-pruning

- Post-pruning examines node ③ and node ① similarly.

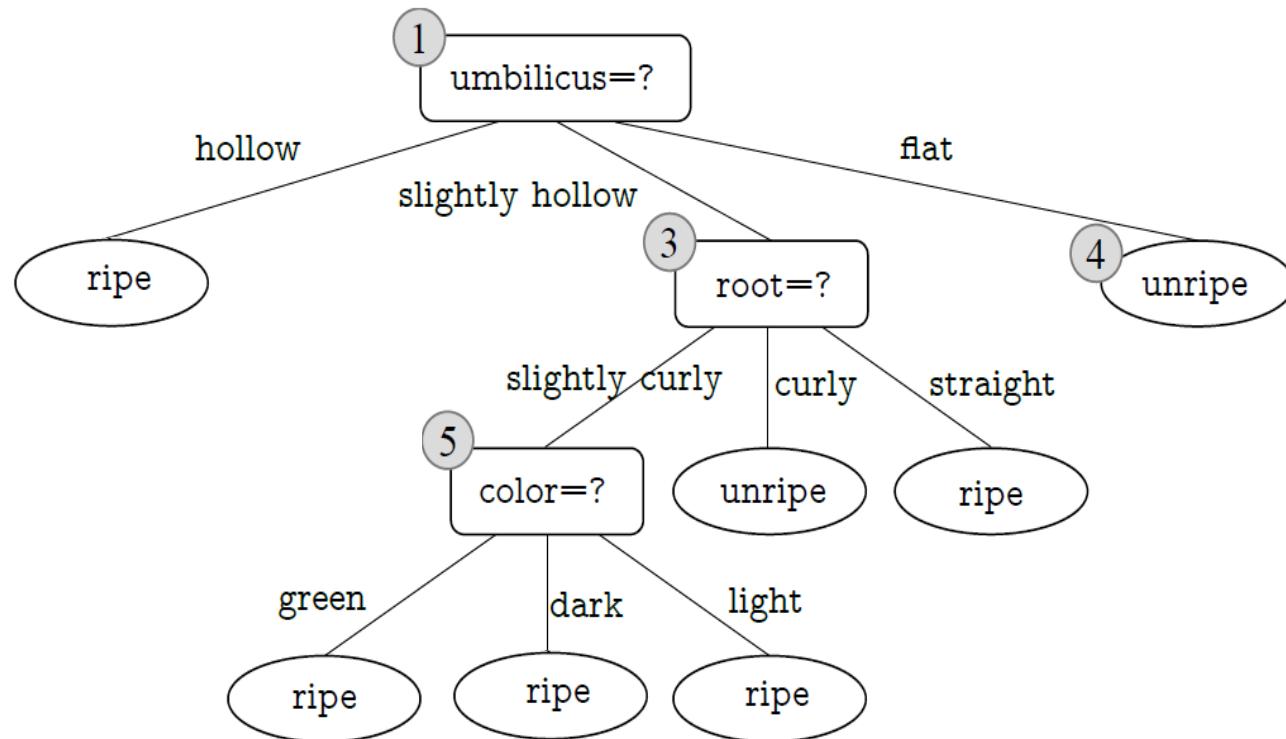


validation accuracy
before pruning: 71.4%
after pruning: 42.9%
decision: no pruning

validation accuracy
before pruning: 71.4%
after pruning: 71.4%
decision: no pruning

Pruning: Post-pruning

- Finally, the post-pruning decision tree is



Pruning: Post-pruning

■ Advantage

- Post-pruning keeps more branches than pre-pruning. In general, post-pruning is **less prone to underfitting** and leads to **better generalization ability** compared to pre-pruning.

■ Disadvantage

- The **training time of post-pruning is much longer** since it takes a bottom-up strategy to examine every non-leaf node in a completely grown decision tree.



Outline

- Basic Process
- Split Selection
- Pruning
- Continuous and Missing Values
- Multivariate Decision Trees



Continuous Values

□ Discretization Strategy (Bi-partition)

- Given a data set D and a continuous feature a , suppose n values of a are observed in D , and we sort these values in ascending order, denoted by a^1, a^2, \dots, a^n . With a split point t , D is partitioned into the subsets D_t^- and D_t^+ , where D_t^- includes the samples with the value of a not greater than t , and D_t^+ includes the samples with the value of a greater than t . There are $n - 1$ elements in the following set of candidate split points:

$$T_a = \left\{ \frac{a^i + a^{i+1}}{2} \mid 1 \leq i \leq n - 1 \right\}$$

where the midpoint $\frac{a^i + a^{i+1}}{2}$ is used as the candidate split point for the interval $[a^i, a^{i+1}]$.

Continuous Values

□ Discretization Strategy (Bi-partition)

- The split points are examined in the same way as discrete features, and the optimal split points are selected for splitting nodes.

$$\begin{aligned}\text{Gain}(D, a) &= \max_{t \in T_a} \text{Gain}(D, a, t) \\ &= \max_{t \in T_a} \text{Ent}(D) - \sum_{\lambda \in \{-, +\}} \frac{|D_t^\lambda|}{|D|} \text{Ent}(D_t^\lambda)\end{aligned}$$

where $\text{Gain}(D, a, t)$ is the information gain of bi-partitioning D by t , and the split point with the largest $\text{Gain}(D, a, t)$ is selected.

Continuous Values

A Concrete Example

ID	color	root	sound	texture	umbilicus	surface	density	sugar	ripe
1	green	curly	muffled	clear	hollow	hard	0.697	0.460	true
2	dark	curly	dull	clear	hollow	hard	0.774	0.376	true
3	dark	curly	muffled	clear	hollow	hard	0.634	0.264	true
4	green	curly	dull	clear	hollow	hard	0.608	0.318	true
5	light	curly	muffled	clear	hollow	hard	0.556	0.215	true
6	green	slightly curly	muffled	clear	slightly hollow	soft	0.403	0.237	true
7	dark	slightly curly	muffled	slightly blurry	slightly hollow	soft	0.481	0.149	true
8	dark	slightly curly	muffed	clear	slightly hollow	hard	0.437	0.211	true
9	dark	slightly curly	dull	slightly blurry	slightly hollow	hard	0.666	0.091	false
10	green	straight	crisp	clear	flat	soft	0.243	0.267	false
11	light	straight	crisp	blurry	flat	hard	0.245	0.057	false
12	light	curly	muffed	blurry	flat	soft	0.343	0.099	false
13	green	slightly curly	muffed	slightly blurry	hollow	hard	0.639	0.161	false
14	light	slightly curly	dull	slightly blurry	hollow	hard	0.657	0.198	false
15	dark	slightly curly	muffed	clear	slightly hollow	soft	0.360	0.370	false
16	light	curly	muffed	blurry	flat	hard	0.593	0.042	false
17	green	curly	dull	slightly blurry	slightly hollow	hard	0.719	0.103	false

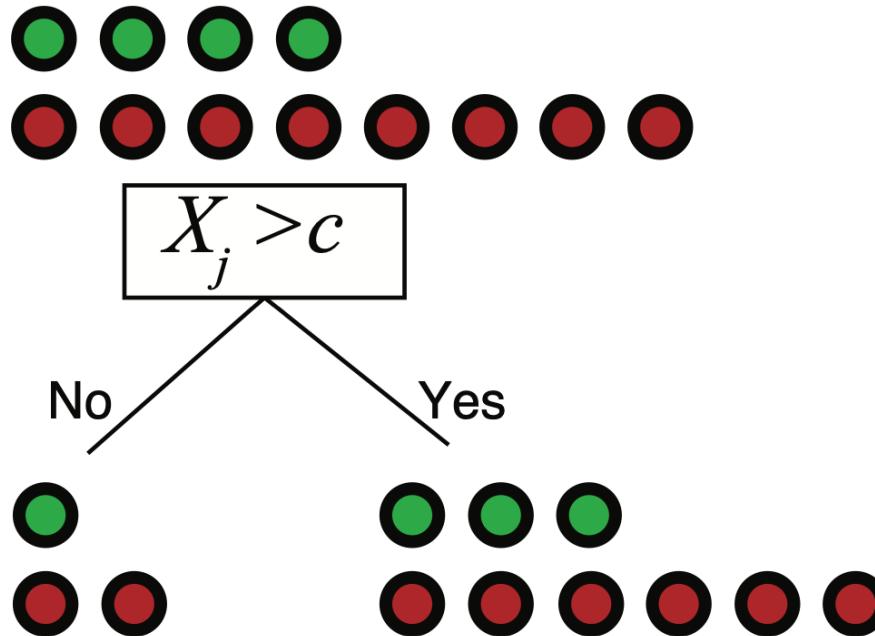
For the feature density, its candidate split point set includes 16 values:

$$T_{\text{density}} = \{0.244, 0.294, 0.351, 0.381, 0.420, 0.459, 0.518, 0.574, 0.600, 0.621, 0.636, 0.648, 0.661, 0.681, 0.708, 0.746\}$$

The information gain of density is 0.262, and the corresponding split point is 0.381.

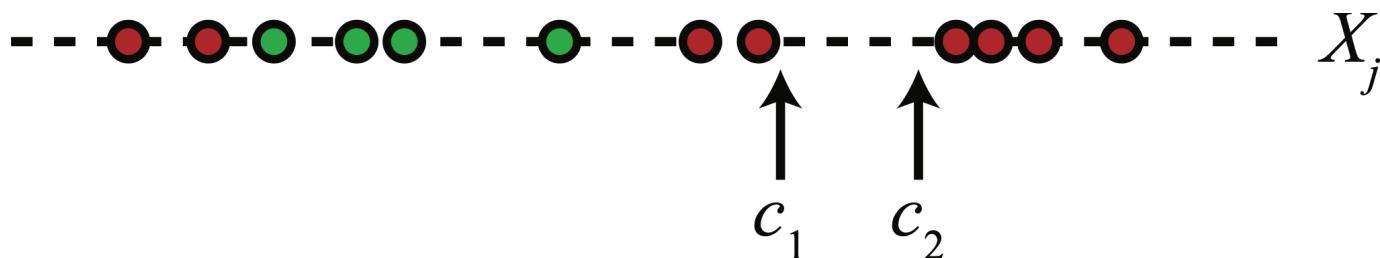
Unlike discrete features, a continuous feature can be used as a splitting feature more than once in a decision sequence.

Optimal splits for continuous attributes

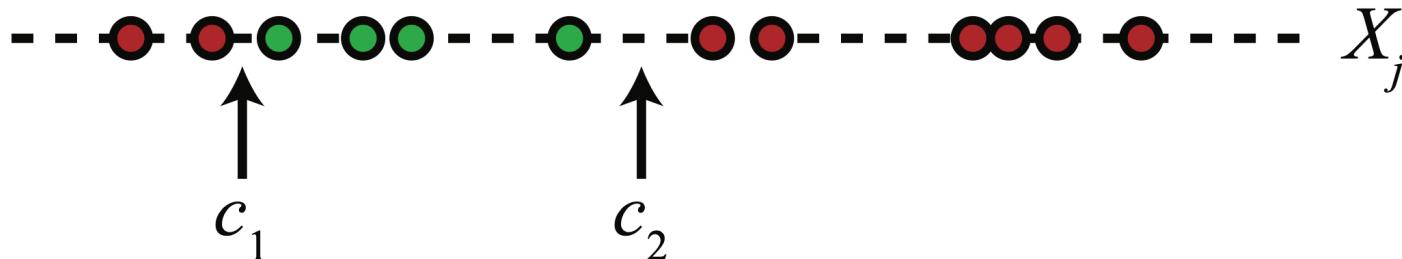


Optimal splits for continuous attributes

- Moving split point along the empty space between two observed values has no effect on information gain or empirical loss; so just use midpoint



- Moreover, only splits between examples from different classes can be optimal for information gain or empirical loss reduction



Missing Values

- In practice, data is often incomplete, that is, some feature values are missing in some samples.
- Can we simply discard the incomplete samples?

It is a huge waste of data.

- Learning from incomplete samples raises two problems:

Q1: how to choose the splitting features when there are missing values

Q2: how to split a sample with the splitting feature value missing?



Missing Values

- Given a training set D and a feature a , let \tilde{D} be the subset of samples in D that have values of a , \tilde{D}^v denote the subset of samples in \tilde{D} taking the value a^v , \tilde{D}_k denote the subset of samples in \tilde{D} belonging to the k th class. We assign a weight w_x to each sample x , and define:

- the proportion of samples without missing values

$$\rho = \frac{\sum_{x \in \tilde{D}} w_x}{\sum_{x \in D} w_x}$$

- the proportion of the k th class in all samples without missing values

$$\tilde{p}_k = \frac{\sum_{x \in \tilde{D}_k} w_x}{\sum_{x \in \tilde{D}} w_x} \quad (1 \leq k \leq |\mathcal{Y}|)$$

- the proportion of samples taking the feature value a^v in all samples without missing values

$$\tilde{r}_v = \frac{\sum_{x \in \tilde{D}^v} w_x}{\sum_{x \in \tilde{D}} w_x} \quad (1 \leq v \leq V)$$

Missing Values

- For Q1, with the above definitions, we extend the information gain to

$$\begin{aligned}\text{Gain}(D, a) &= \rho \times \text{Gain}(\tilde{D}, a) \\ &= \rho \times \left(\text{Ent}(\tilde{D}) - \sum_{v=1}^V \tilde{r}_v \text{Ent}(\tilde{D}^v) \right)\end{aligned}$$

where

$$\text{Ent}(\tilde{D}) = - \sum_{k=1}^{|\mathcal{Y}|} \tilde{p}_k \log_2 \tilde{p}_k$$

- For Q2,
 - when the value of a is known, we place the sample x into the corresponding child node without changing its weight w_x .
 - when the value of a is unknown, we place the sample x into all child nodes, and set its weight in the child node of value a^v to $\tilde{r}_v \cdot w_x$. In other words, we place the same sample into different child nodes with different probabilities.

Missing Values

A Concrete Example

ID	color	root	sound	texture	umbilicus	surface	ripe
1	-	curly	muffled	clear	hollow	hard	true
2	dark	curly	dull	clear	hollow	-	true
3	dark	curly	-	clear	hollow	hard	true
4	green	curly	dull	clear	hollow	hard	true
5	-	curly	muffled	clear	hollow	hard	true
6	green	slightly curly	muffled	clear	-	soft	true
7	dark	slightly curly	muffled	slightly blurry	slightly hollow	soft	true
8	dark	slightly curly	muffled	-	slightly hollow	hard	true
9	dark	-	dull	slightly blurry	slightly hollow	hard	false
10	green	straight	crisp	-	flat	soft	false
11	light	straight	crisp	blurry	flat	-	false
12	light	curly	-	blurry	flat	soft	false
13	-	slightly curly	muffled	slightly blurry	hollow	hard	false
14	light	slightly curly	dull	slightly blurry	hollow	hard	false
15	dark	slightly curly	muffled	clear	-	soft	false
16	light	curly	muffled	blurry	flat	hard	false
17	green	-	dull	slightly blurry	slightly hollow	hard	false

- In the beginning, the root node includes all of the 17 samples in D , and all samples have the weight of 1.
- Taking color as an example, the set of samples without missing values of this feature, denoted by \tilde{D} , includes 14 samples. The entropy of \tilde{D} is calculated as

$$\text{Ent}(\tilde{D}) = - \sum_{k=1}^2 \tilde{p}_k \log_2 \tilde{p}_k = -\left(\frac{6}{14} \log_2 \frac{6}{14} + \frac{8}{14} \log_2 \frac{8}{14}\right) = 0.985$$

Missing Values

- Let \tilde{D}^1 , \tilde{D}^2 , and \tilde{D}^3 be the subsets of samples with color = green, color = dark, and color = light, respectively. Then, we have

$$\text{Ent}(\tilde{D}^1) = -\left(\frac{2}{4} \log_2 \frac{2}{4} + \frac{2}{4} \log_2 \frac{2}{4}\right) = 1.000 \quad \text{Ent}(\tilde{D}^2) = -\left(\frac{4}{6} \log_2 \frac{4}{6} + \frac{2}{6} \log_2 \frac{2}{6}\right) = 0.918$$

$$\text{Ent}(\tilde{D}^3) = -\left(\frac{0}{4} \log_2 \frac{0}{4} + \frac{4}{4} \log_2 \frac{4}{4}\right) = 0.000$$

- The information gain of color for subset \tilde{D} is

$$\begin{aligned}\text{Gain}(\tilde{D}, \text{color}) &= \text{Ent}(\tilde{D}) - \sum_{v=1}^3 \tilde{r}_v \text{Ent}(\tilde{D}^v) \\ &= 0.985 - \left(\frac{4}{14} \times 1.000 + \frac{6}{14} \times 0.918 + \frac{4}{14} \times 0.000 \right) \\ &= 0.306.\end{aligned}$$

- The information gain of color for data set D is

$$\text{Gain}(D, \text{color}) = \rho \times \text{Gain}(\tilde{D}, \text{color}) = \frac{14}{17} \times 0.306 = 0.252.$$

Missing Values

■ Similarly, we have

$$\text{Gain}(D, \text{color}) = 0.252;$$

$$\text{Gain}(D, \text{sound}) = 0.252;$$

$$\text{Gain}(D, \text{umbilicus}) = 0.289;$$

$$\text{Gain}(D, \text{root}) = 0.171;$$

$$\text{Gain}(D, \text{texture}) = 0.424;$$

$$\text{Gain}(D, \text{texture}) = 0.006.$$

clear

slightly blurry

blurry

The weights of these samples (i.e., 1) remain unchanged in the child nodes.

Missing. The sample is placed into all of the three child nodes with different weights:

$\frac{7}{15}$, $\frac{5}{15}$, and $\frac{3}{15}$.

ID	color	root	sound	texture	umbilicus	surface	ripe
1	-	curly	muffled	clear	hollow	hard	true
2	dark	curly	dull	clear	hollow	-	true
3	dark	curly	-	clear	hollow	hard	true
4	green	curly	dull	clear	hollow	hard	true
5	-	curly	muffled	clear	hollow	hard	true
6	green	slightly curly	muffled	clear	-	soft	true
7	dark	slightly curly	muffled	slightly blurry	slightly hollow	soft	true
8	dark	slightly curly	muffled	-	slightly hollow	hard	true
9	dark	-	dull	slightly blurry	slightly hollow	hard	false
10	green	straight	crisp	-	flat	soft	false
11	light	straight	crisp	blurry	flat	-	false
12	light	curly	-	blurry	flat	soft	false
13	-	slightly curly	muffled	slightly blurry	hollow	hard	false
14	light	slightly curly	dull	slightly blurry	hollow	hard	false
15	dark	slightly curly	muffled	clear	-	soft	false
16	light	curly	muffled	blurry	flat	hard	false
17	green	-	dull	slightly blurry	slightly hollow	hard	false



Missing Values

■ Any other approaches?

Imputation of missing values

```
>>> import numpy as np
>>> from sklearn.impute import SimpleImputer
>>> imp = SimpleImputer(missing_values=np.nan, strategy='mean')
>>> imp.fit([[1, 2], [np.nan, 3], [7, 6]])
SimpleImputer()
>>> X = [[np.nan, 2], [6, np.nan], [7, 6]]
>>> print(imp.transform(X))
[[4.          2.        ]
 [6.          3.666...]
 [7.          6.        ]]
```

strategy : str, default='mean'

The imputation strategy.

- If "mean", then replace missing values using the mean along each column. Can only be used with numeric data.
- If "median", then replace missing values using the median along each column. Can only be used with numeric data.
- If "most_frequent", then replace missing using the most frequent value along each column. Can be used with strings or numeric data. If there is more than one such value, only the smallest is returned.
- If "constant", then replace missing values with fill_value. Can be used with strings or numeric data.

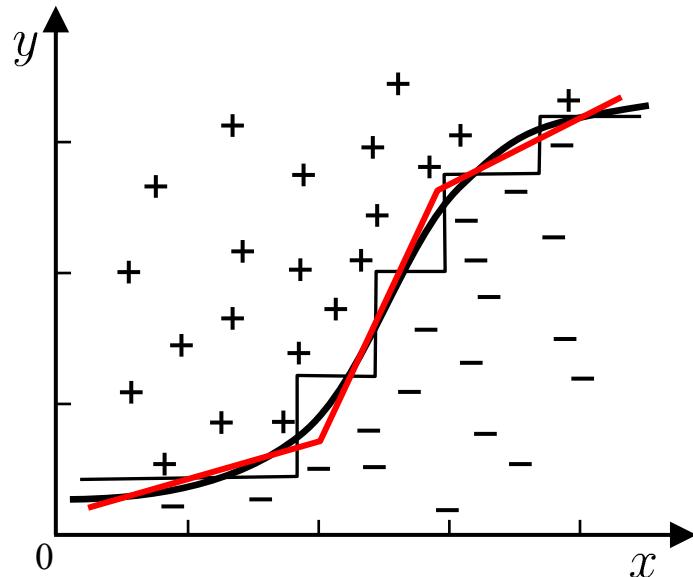
Outline

- Basic Process
- Split Selection
- Pruning
- Continuous and Missing Values
- Multivariate Decision Trees



Multivariate Decision Trees

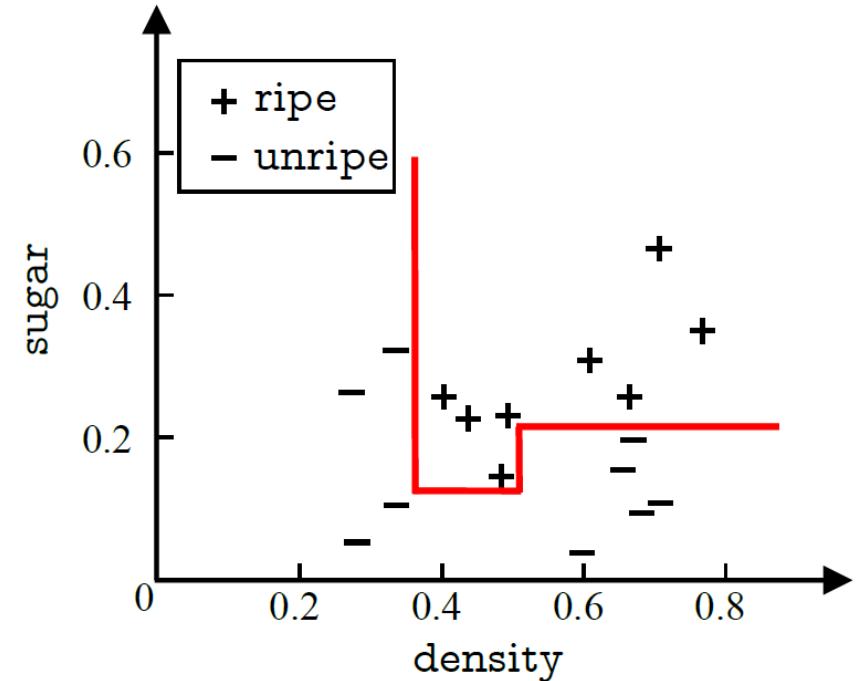
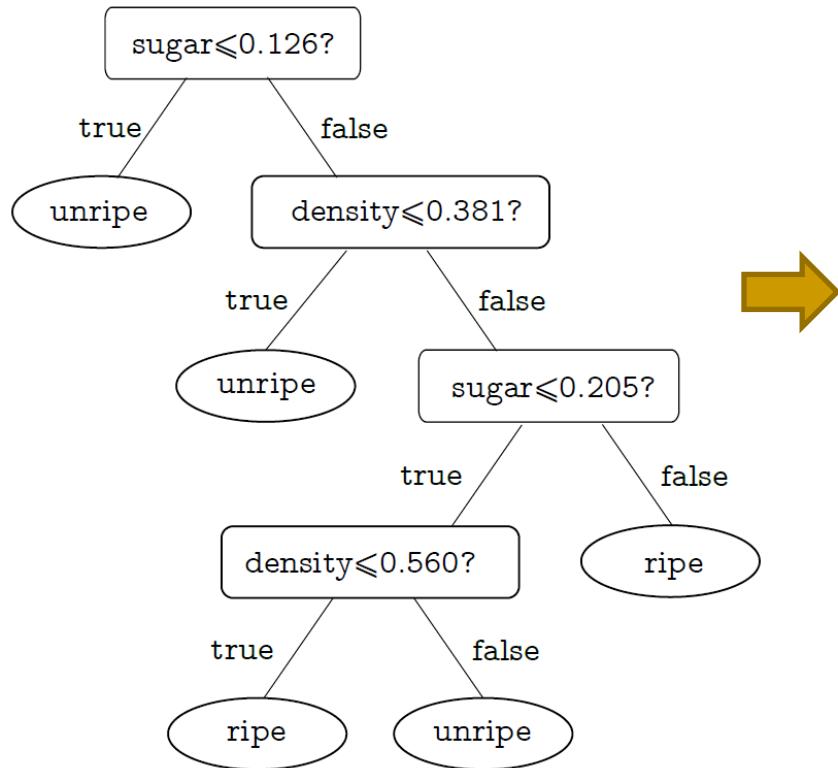
- For decision trees, the decision boundaries are axis-parallel.
- *Multivariate Decision Tree*



- Each non-leaf node is no longer a test for a particular feature but a linear combination of features.
- ↓
- Each non-leaf node is a linear classifier in the form of $\sum_{i=1}^d w_i a_i = t$, where w_i is the weight of feature a_i , and w_i and t are learned from the data set and feature set of the node.

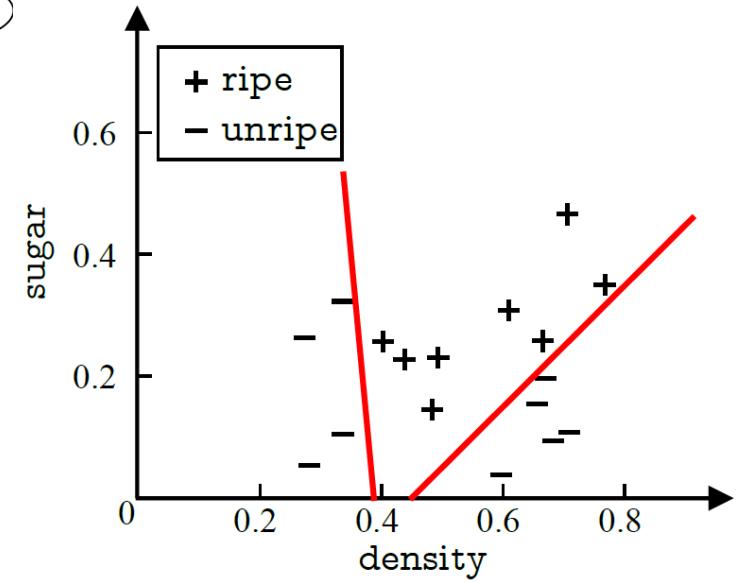
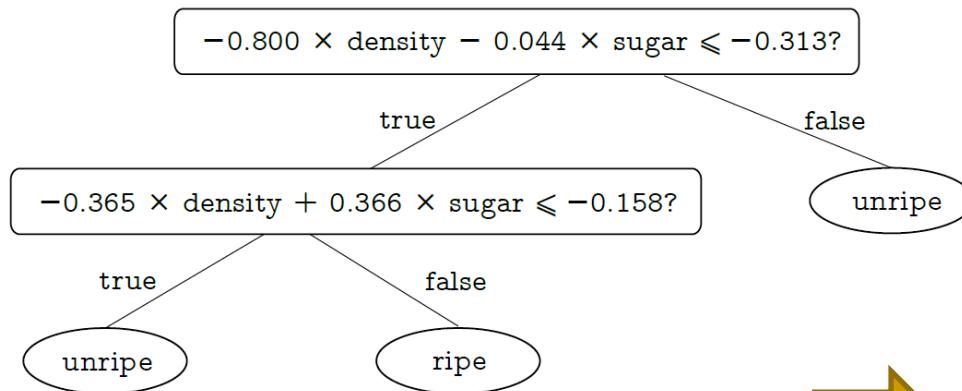
Multivariate Decision Trees

■ Decision Tree



Multivariate Decision Trees

■ Multivariate Decision Trees



Take Home Message

- Splitting Features Selection
- Pruning (Pre-pruning and Post-pruning)
- Continuous and Missing Values
- Multivariate Decision Tree

Software Packages

- Scikit-learn

<https://scikit-learn.org/stable/modules/tree.html>

- ID₃, C_{4.5}, C_{5.0}

<http://www.rulequest.com/Personal/>

- J₄₈

<http://www.cs.waikato.ac.nz/ml/weka/>



feature importance

```
...056 cpdef compute_feature_importances(self, normalize=True):
1057     """Computes the importance of each feature (aka variable)."""
1058     cdef Node* left
1059     cdef Node* right
1060     cdef Node* nodes = self.nodes
1061     cdef Node* node = nodes
1062     cdef Node* end_node = node + self.node_count
1063
1064     cdef double normalizer = 0.
1065
1066     cdef np.ndarray[np.float64_t, ndim=1] importances
1067     importances = np.zeros((self.n_features,))
1068     cdef DOUBLE_t* importance_data = <DOUBLE_t*>importances.data
1069
1070     with nogil:
1071         while node != end_node:
1072             if node.left_child != _TREE_LEAF:
1073                 # ... and node.right_child != _TREE_LEAF:
1074                 left = &nodes[node.left_child]
1075                 right = &nodes[node.right_child]
1076
1077                 importance_data[node.feature] += (
1078                     node.weighted_n_node_samples * node.impurity -
1079                     left.weighted_n_node_samples * left.impurity -
1080                     right.weighted_n_node_samples * right.impurity)
1081             node += 1
1082
1083             importances /= nodes[0].weighted_n_node_samples
1084
1085             if normalize:
1086                 normalizer = np.sum(importances)
1087
1088             if normalizer > 0.0:
1089                 # Avoid dividing by zero (e.g., when root is pure)
1090                 importances /= normalizer
1091
1092     return importances
```

