Classification

Machine Learning

Decision Tree Induction

Classification for Predictive Analysis

Classification is the process of finding a model (or function) that describes and distinguishes data classes or concepts.

The model are derived based on the analysis of a set of training data (i.e., data objects for which the class labels are known).

The model is used to predict the class label of objects for which the class label is unknown.

Classification for Predictive Analysis

“How is the derived model presented?”

The derived model may be represented in various forms, such as

classification rules (i.e., IF-THEN rules),

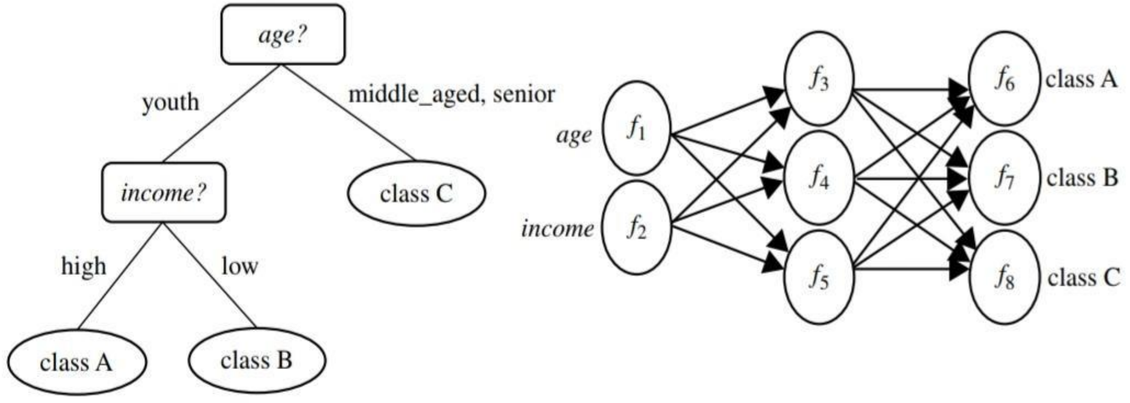
decision trees,

mathematical formulae, or

neural networks.

age(*X*, “youth”) *AND income*(*X*, “high”) $\longrightarrow$ *class*(*X*, “A”)
age(*X*, “youth”) *AND income*(*X*, “low”) $\longrightarrow$ *class*(*X*, “B”)
age(*X*, “middle_aged”) $\longrightarrow$ *class*(*X*, “C”)
age(*X*, “senior”) $\longrightarrow$ *class*(*X*, “C”)

(a)



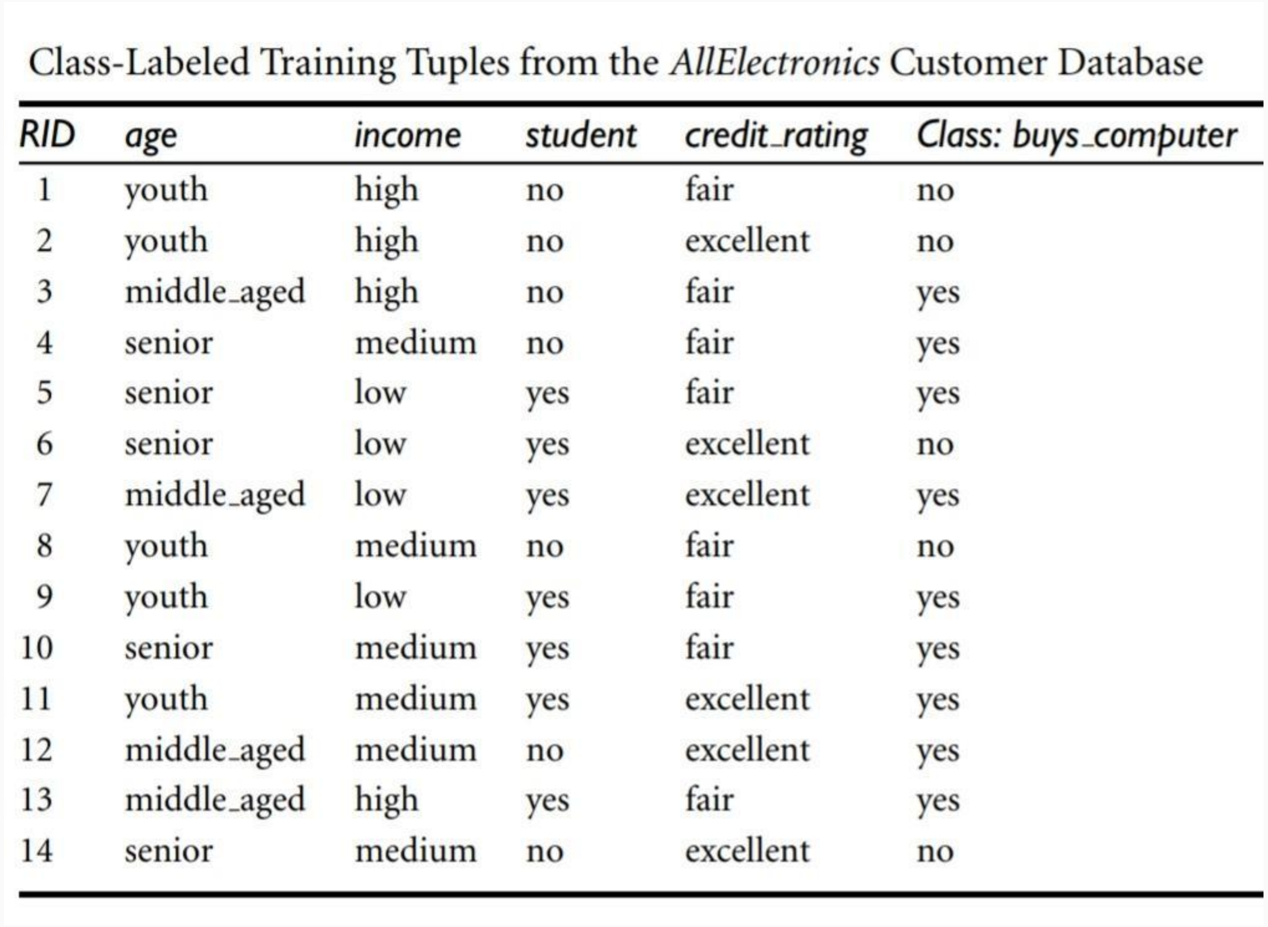
(b) **(c)**

Decision tree induction

Decision tree induction is the learning of decision trees from class-labeled training tuples.

A **decision tree** is a flowchart-like tree structure, where each internal node (non leaf node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (or terminal node) holds a class label.

Decision tree induction algorithms: **ID3, C4.5, and CART**

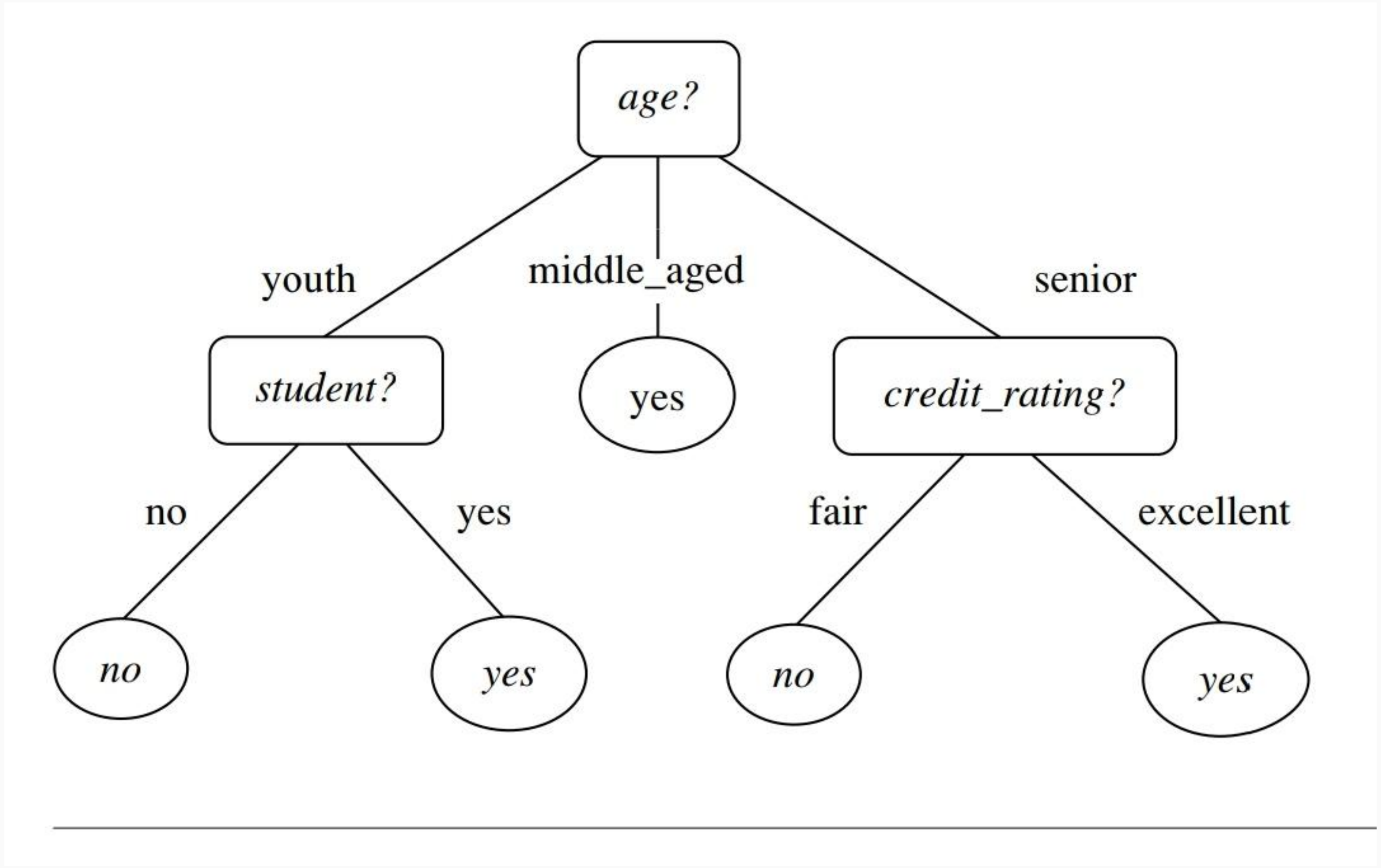


Class-Labeled Training Tuples from the *AllElectronics* Customer Database

RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Dataset

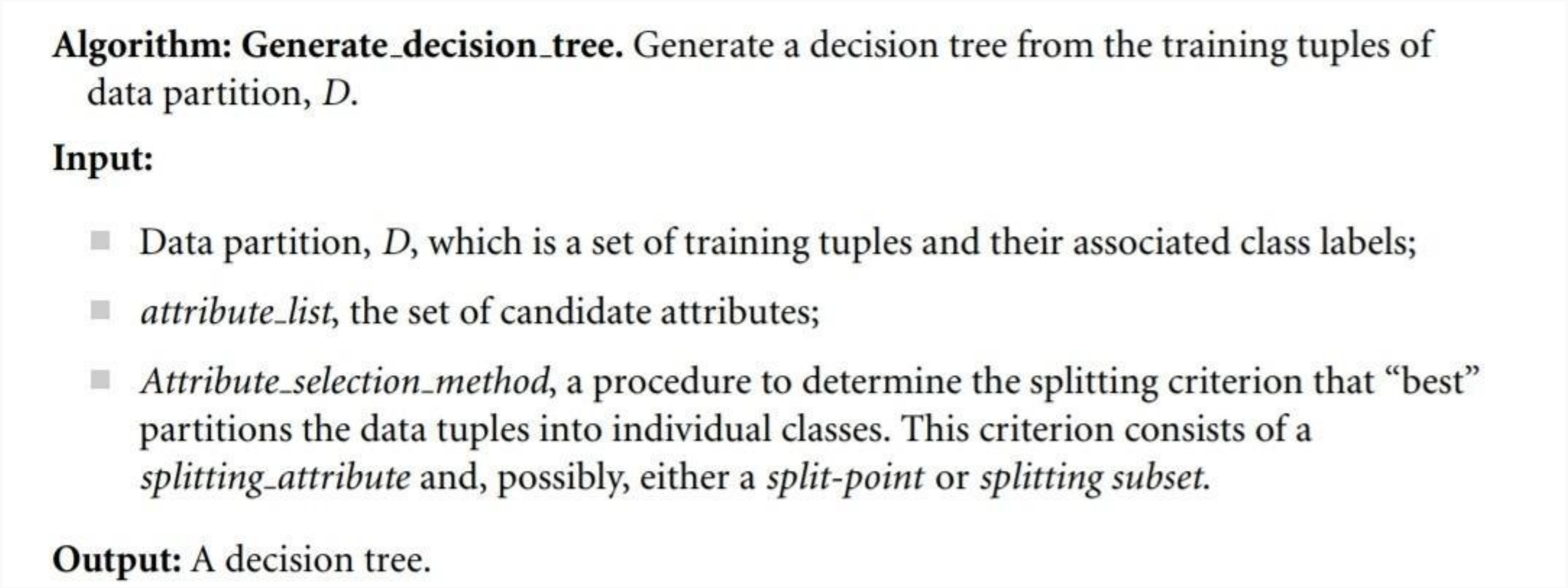
Decision tree induction



Decision tree induction:
“Why are decision tree classifiers so popular?”

- → does not require any domain knowledge or parameter setting.
- → can handle multidimensional data.
- → steps of decision tree induction are simple and fast and have good accuracy
- → many application areas such as medicine, manufacturing and production, financial analysis, astronomy, and molecular biology.

Basic algorithm for inducing a decision tree from training tuples.



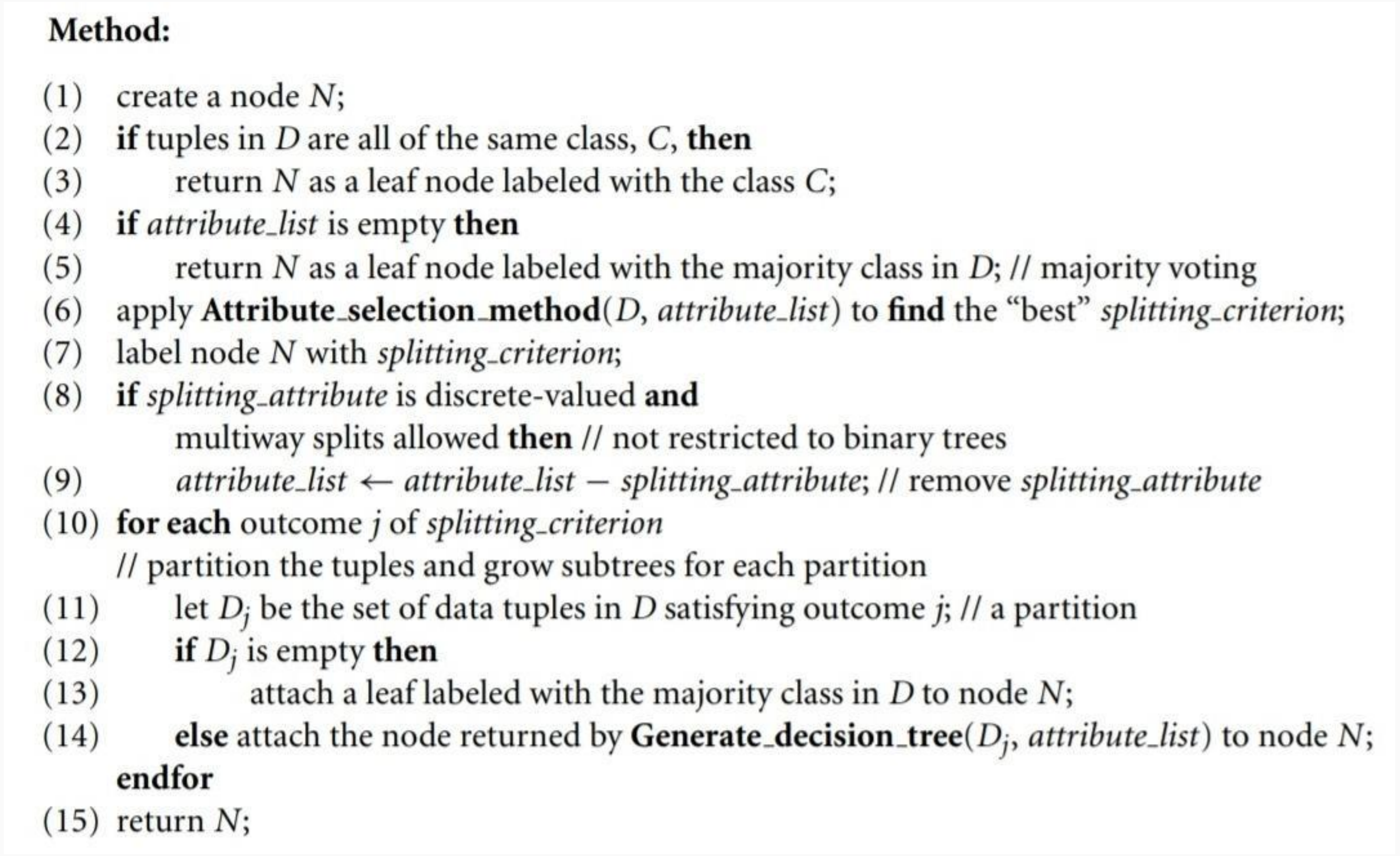
Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition, *D*.

Input:

- Data partition, *D*, which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a *splitting_attribute* and, possibly, either a *split-point* or *splitting subset*.

Output: A decision tree.

Basic algorithm for inducing a decision tree from training tuples.



Method:

```
(1)  create a node $N$;
(2)  if tuples in $D$ are all of the same class, $C$, then
(3)      return $N$ as a leaf node labeled with the class $C$;
(4)  if attribute_list is empty then
(5)      return $N$ as a leaf node labeled with the majority class in $D$; // majority voting
(6)  apply Attribute_selection_method($D$, attribute_list) to find the "best" splitting_criterion;
(7)  label node $N$ with splitting_criterion;
(8)  if splitting_attribute is discrete-valued and
         multiway splits allowed then // not restricted to binary trees
(9)      attribute_list $\leftarrow$ attribute_list $-$ splitting_attribute; // remove splitting_attribute
(10) for each outcome $j$ of splitting_criterion
     // partition the tuples and grow subtrees for each partition
(11)     let $D_j$ be the set of data tuples in $D$ satisfying outcome $j$; // a partition
(12)     if $D_j$ is empty then
(13)         attach a leaf labeled with the majority class in $D$ to node $N$;
(14)     else attach the node returned by Generate_decision_tree($D_j$, attribute_list) to node $N$;
     endfor
(15) return $N$;
```

Decision tree induction: “Attribute Selection Measures”

An **attribute selection measure** is a heuristic for selecting the splitting criterion that “best” separates a given data partition, D, of class-labeled training tuples into individual classes.

If we were to split D into smaller partitions according to the outcomes of the splitting criterion, ideally each partition would be **pure** (i.e., all the tuples that fall into a given partition would belong to the same class).

Three popular attribute selection measures—**information gain, gain ratio, and Gini index.**

Decision tree induction: “Attribute Selection Measures: Information gain”

ID3 uses **information gain** as its attribute selection measure.

The attribute with the highest information gain is chosen as the splitting attribute.

It tries to make the partition as **pure** as possible.

Info(D) is also known as the entropy of D.

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i),$$

Decision tree induction: “Attribute Selection Measures: Information gain”

suppose we were to partition the tuples in D on some attribute A having v distinct values, {a 1 , a 2 , . . . , a v }, as observed from the training data.

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j).$$

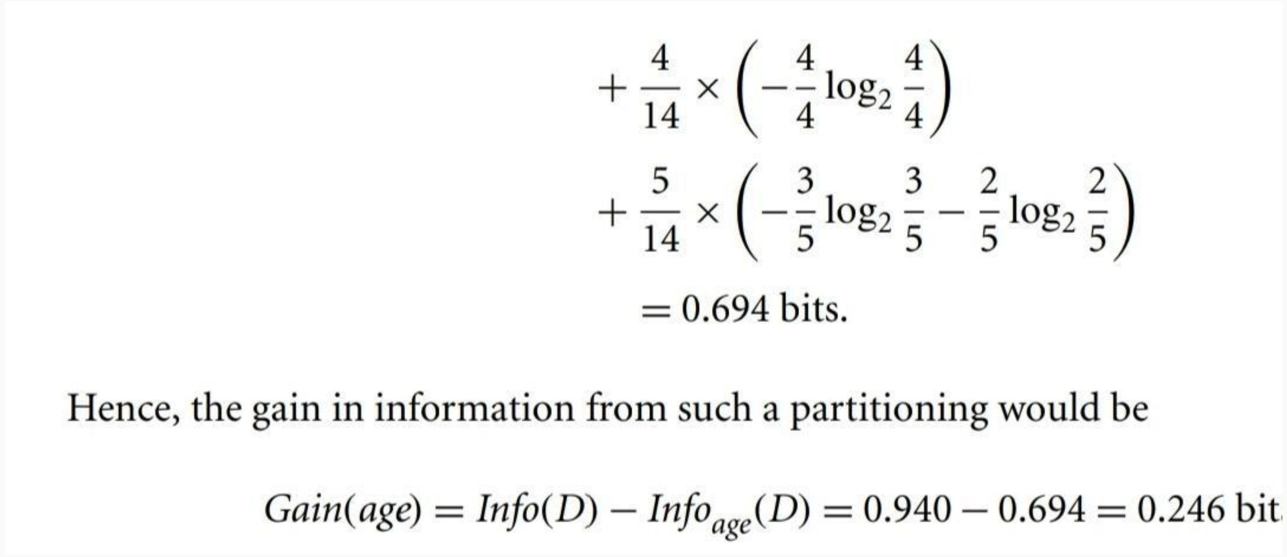
$$Gain(A) = Info(D) - Info_A(D).$$

Decision tree induction: “Attribute Selection Measures: Information gain”

$$Info(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits.}$$

$$Info_{age}(D) = \frac{5}{14} \times \left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right)$$

Decision tree induction:
“Attribute Selection Measures: Information gain”



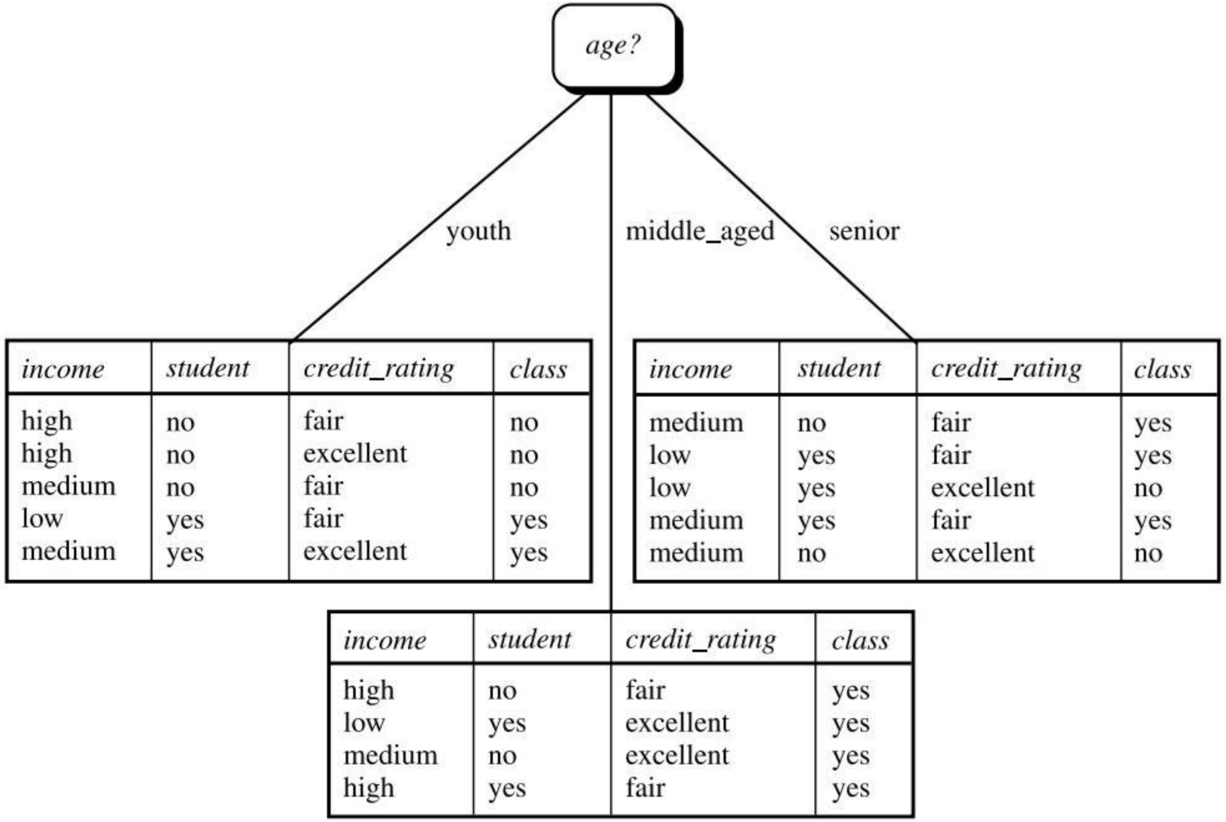
$$+\frac{4}{14}\times\left(-\frac{4}{4}\log_2\frac{4}{4}\right)$$

$$+\frac{5}{14}\times\left(-\frac{3}{5}\log_2\frac{3}{5}-\frac{2}{5}\log_2\frac{2}{5}\right)$$

$$=0.694 \text{ bits.}$$

Hence, the gain in information from such a partitioning would be

$$Gain(age) = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 \text{ bit}$$



income	*student*	*credit_rating*	*class*
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	*student*	*credit_rating*	*class*
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

income	*student*	*credit_rating*	*class*
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

The attribute age has the highest information gain

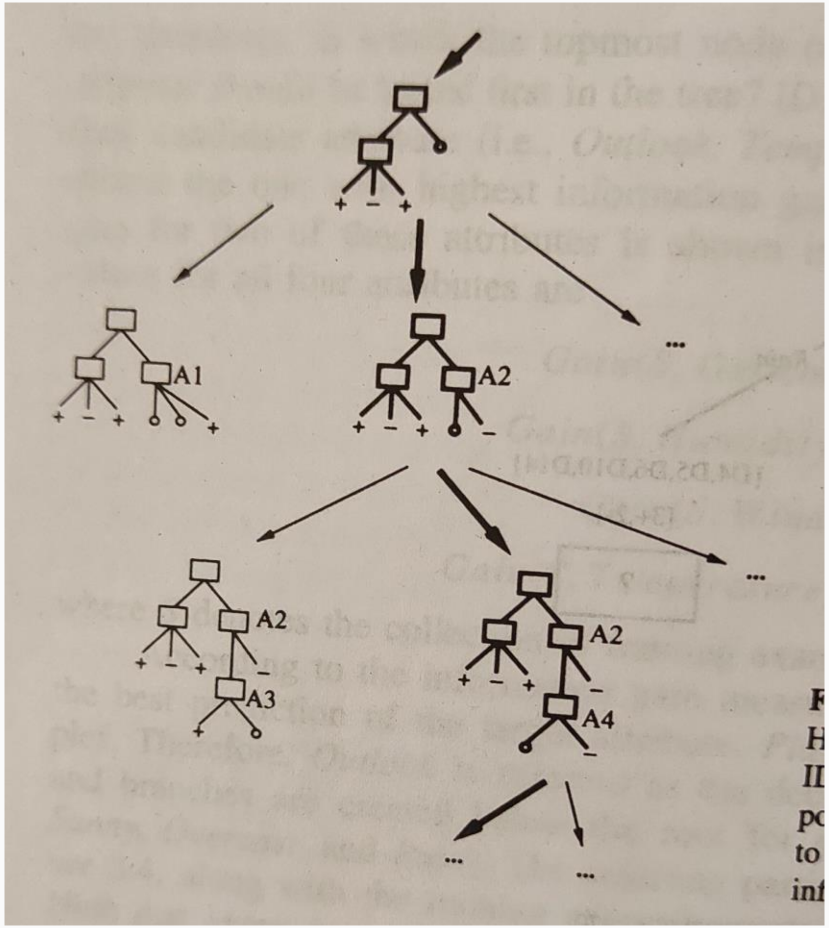
Decision tree induction: “Appropriate problems for Decision Tree Learning”

- Instances are represented by attribute-value pairs.
- The target function has discrete output values
- Disjunctive descriptions may be required: [(P^Q)V(~P ^ Q) V (~P ^ ~Q)]
- The training data may contain errors
- The training data may contain missing attribute values.

Decision tree induction:
“Hypothesis space search in decision tree learning”

ID3 searches a space of hypotheses for one that fits the training examples.

It performs a simple-to-complex, hill-climbing search through this hypothesis space, beginning with the empty tree, then considering more elaborate hypotheses.



Hypothesis space search in decision tree learning

Decision tree induction: “Hypothesis space search in decision tree learning”

- ID3’s hypothesis space of all decision trees is a **complete** space of finite discrete-valued functions, relative to the available attributes.
- ID3 maintains only a single current hypothesis as it searches through the space of decision trees. It does not have the ability to determine how many alternative DTs that are consistent with the available training data.

Decision tree induction:
“Hypothesis space search in decision tree learning”

- ID3 in its pure form performs no backtracking in its search. A locally optimal solution corresponds to the decision tree it selects along the single search path it explores.
- ID3 uses all training examples at each step in the search to make statistically based decisions regarding how to refine its current hypothesis. The resulting search is much less sensitive to errors in the individual training examples.

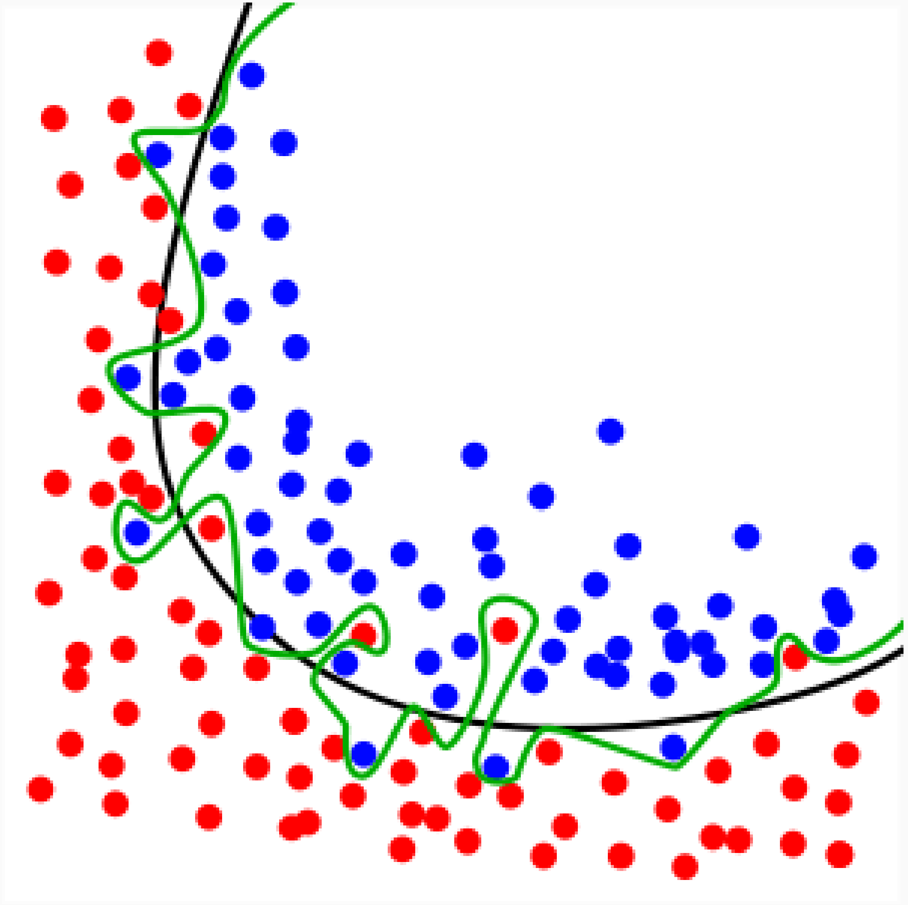
Issues in Decision Tree Learning

1. **Avoid overfitting the data**
   a. Decision tree algorithm grows each branch of the tree just deeply enough to perfectly classify the training examples.
   b. It can lead to difficulties when there is noise in the or when the number of training example is too small.
   c. **Overfitting** refers to a model that models the training data too well. Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data.

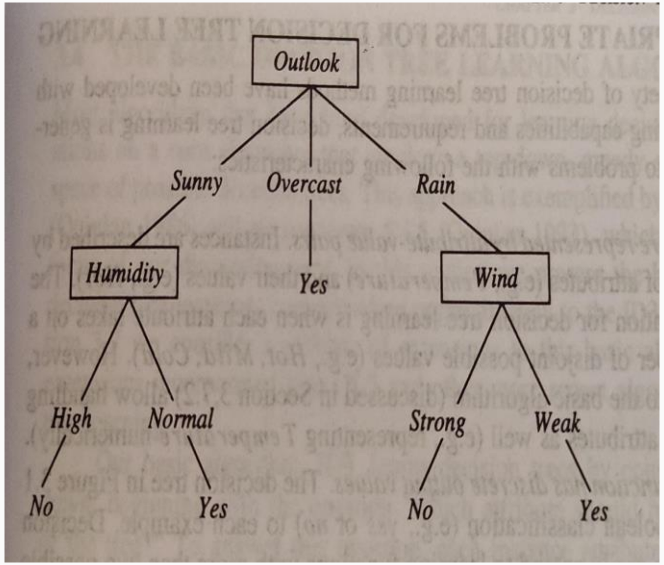
Overfitting

The green line represents an overfitted model and the black line represents a regularized model.

While the green line best follows the training data, it is too dependent on that data and it is likely to have a higher error rate on new unseen data, compared to the black line.



Avoid overfitting the data



(outlook = sunny, Temperature = Hot, Humidity = Normal, Wind = Strong, PlayTennis = **No**)

Incorrect example will now cause ID3 to construct a more complex tree.

Approaches to avoid overfitting

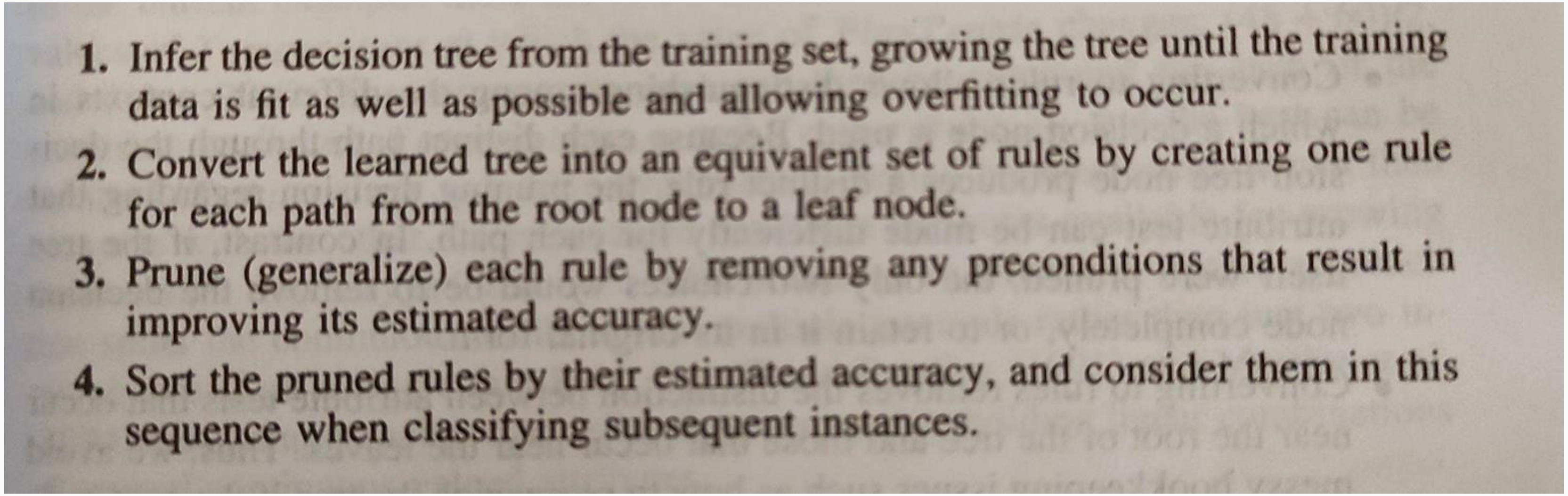
Approaches that stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data.

Approaches that allow the tree to overfit the data, and then post-prune the tree.

Use a seperate set of examples, distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree.

Apply statistical test to estimate whether expanding (or pruning) a particular node (chi-square test).

Rule post-pruning



1. Infer the decision tree from the training set, growing the tree until the training data is fit as well as possible and allowing overfitting to occur.
2. Convert the learned tree into an equivalent set of rules by creating one rule for each path from the root node to a leaf node.
3. Prune (generalize) each rule by removing any preconditions that result in improving its estimated accuracy.
4. Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances.

Rule post-pruning

If (Outlook = Sunny) ^ (Humidity = High)

Then PlayTennis = No

Preconditions: (Outlook = Sunny) & (Humidity = High)

Incorporating Continuous-Valued Attributes

Temperature: 40 48 60 72 80 90

PlayTennis: No No Yes Yes Yes No

Sorting the examples according to the continuous attribute A, then identifying adjacent examples that differ in their target classification.

(48+60)/2 and (80+90)/2: the information gain can then be computed for each candidate attributes. $\text{Temperature}_{>54}$ and $\text{Temperature}_{>85}$ then best can be selected.($\text{Temperature}_{>54}$)