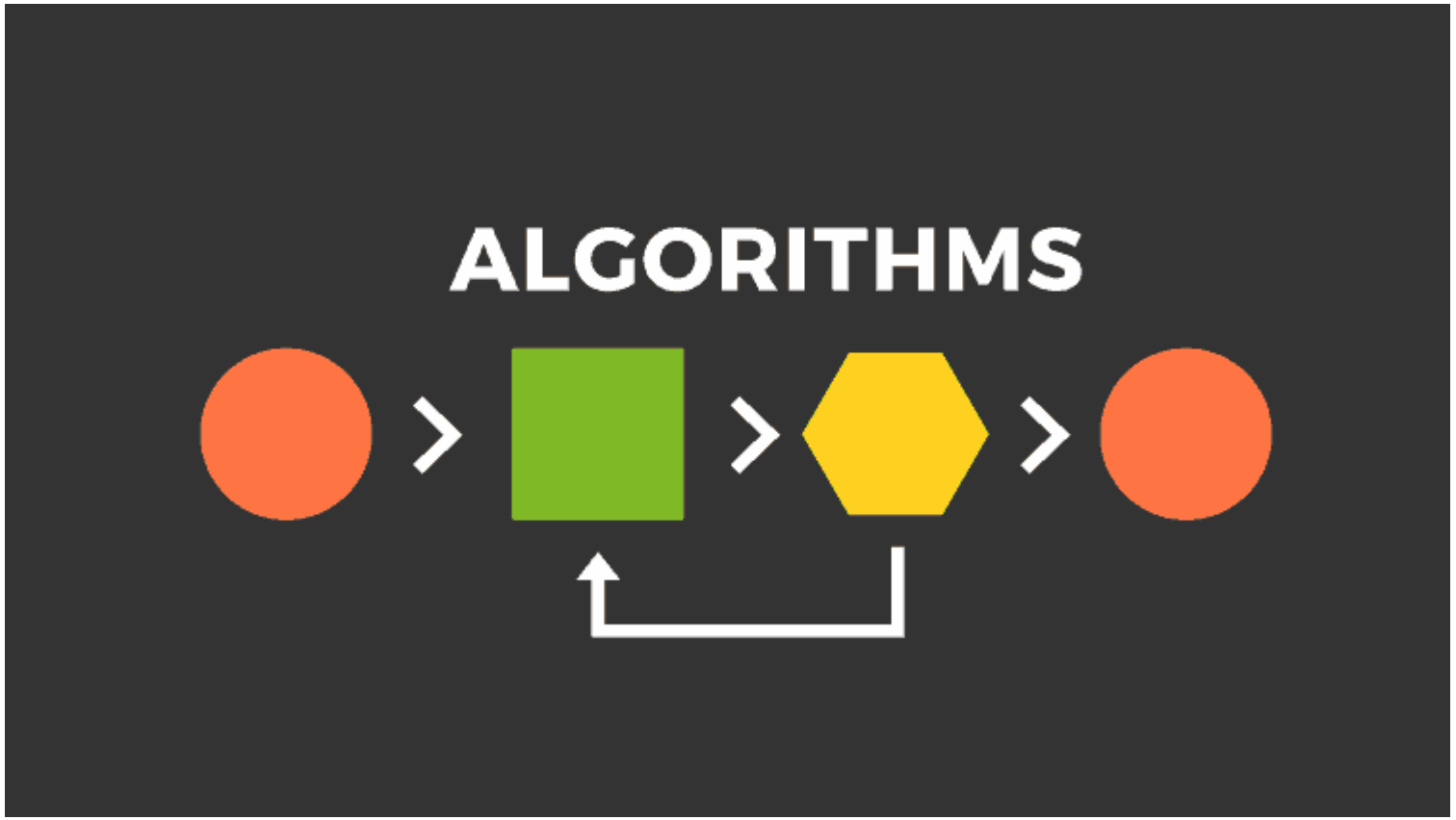
Decision Tree Classification


ALGORITHMS

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 ← 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 ;

Description

- Three parameters: D(Data partition), attribute list, and Attribute selection method
- Initially, **D:** Complete set of training tuples and their associated class labels
- **attribute list:** list of attributes describing the tuples
- **Attribute selection method:** specifies a heuristic procedure for selecting the attribute that “best” discriminates the given tuples according to class

1) create a node N ;
2) **if tuples in D are all of the same class, C, then**
3) **return**
4) **if attribut**
5) **return** **ity 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 ← 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 ;

Tree starts as a single node, N,
representing the training tuples in D

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
8) if splitting
// not restricted to binary trees
9) attribute list ← 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 ;

Termination Condition - 1

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
8) if splitting
// not restricted to binary trees
9) attribute list ← 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 ;

Termination Condition - 2

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)
8)
9)
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)
14)
15) return N ;

Calls Attribute selection method to determine the splitting criterion.
Splitting criterion tells us which attribute to test at node N
by determining the “best” way
to separate or partition the tuples in D into individual classes

It also tells us which branches to grow from node N with respect to the outcomes of the chosen test.
specifically, the splitting criterion indicate either a split-point or a splitting subset.
It is determined so that, ideally, the resulting partitions at each branch are as “pure” as possible.
A partition is pure if all the tuples in it belong to the same class.

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) node N is labeled with the splitting criterion, which serves as a test at the node
9)
10) // 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 ;

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)
8) branch is grown from node N for each of the outcomes of the splitting criterion.
The tuples in D are partitioned accordingly
9)
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 ;

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 ← attribute list − splitting attribute; // remove splitting attribute
10)
Termination Condition - 3
11)
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 ;

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 ← 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)
12)
13)

The algorithm uses the same process recursively to form a decision tree for the tuples at each resulting partition, D j , of D

14) else attach the node returned by Generate decision tree(D j , attribute list) to node N ;
endfor
15) return N ;

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 ← 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)
14) ;
15) return N ;

The resulting decision tree is returned

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 ← 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 ;

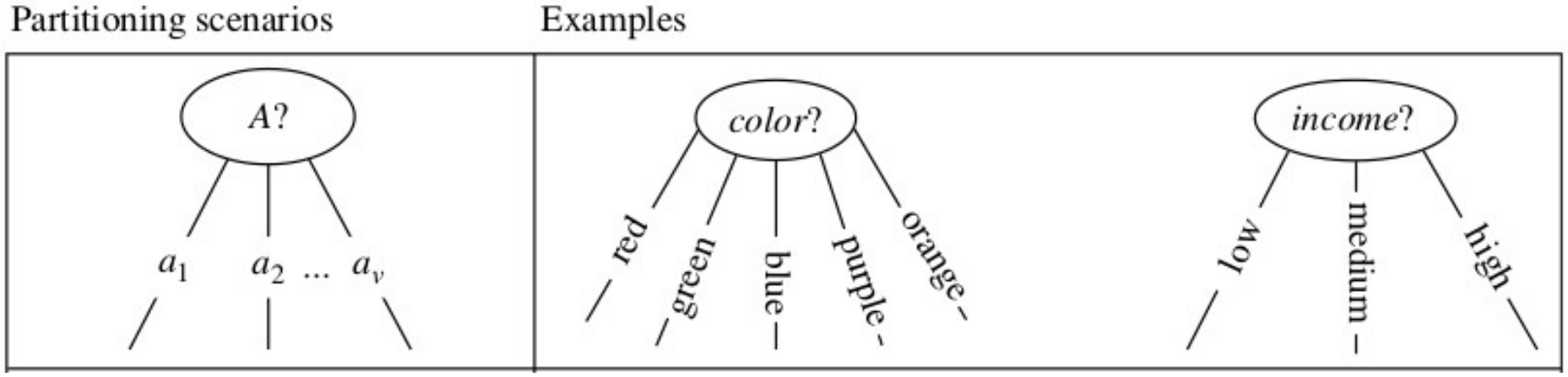
Splitting :Three possible scenarios

1) A is discrete-valued: The outcomes of the test at node N correspond directly to the known values of A.

A branch is created for each known value, aj ,of A and labeled

Partition Dj is subset of class-labeled tuples in D having value aj

Because all the tuples in a given partition have the same value for A, A need not be considered in any future partitioning of the tuples. Therefore, it is removed from attribute list.(Step 8-9)



Splitting :Three possible scenarios

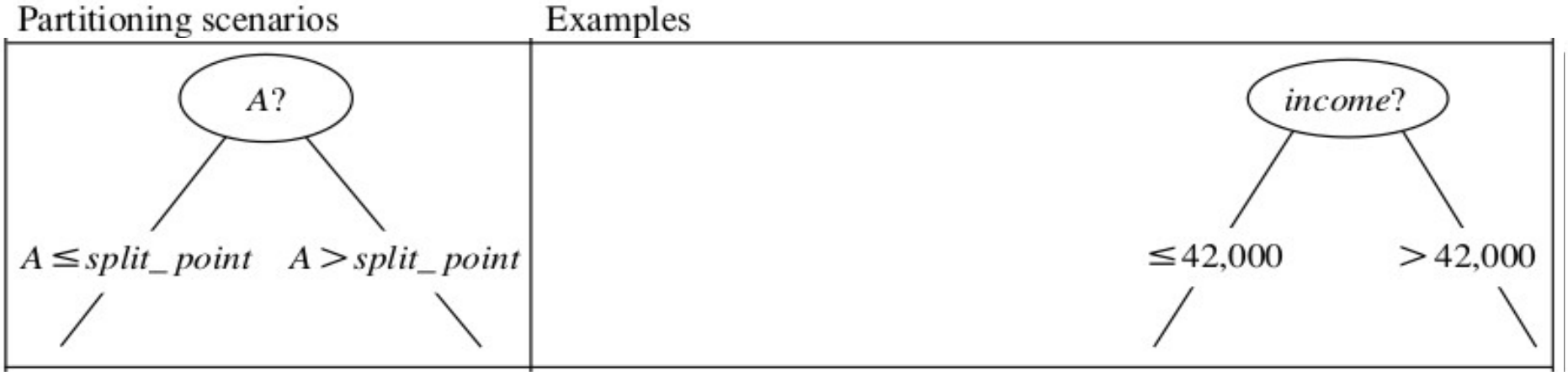
2) A is continuous-valued: In this case, the test at node N has two possible outcomes, corresponding to the conditions A ≤ split point and A > split point, respectively,

where split point is the point returned by ***Attribute selection method*** as part of the splitting criterion

(In practice, the split-point, a, is often taken as the midpoint of two known adjacent values of A and therefore may not actually be a pre-existing value of A from the training data.)

Two branches are grown from N and labelled according to the previous outcomes.

The tuples are partitioned such that D1 holds the subset of class-labelled tuples in D for which A ≤ split point, while D2 holds the rest.



Splitting :Three possible scenarios

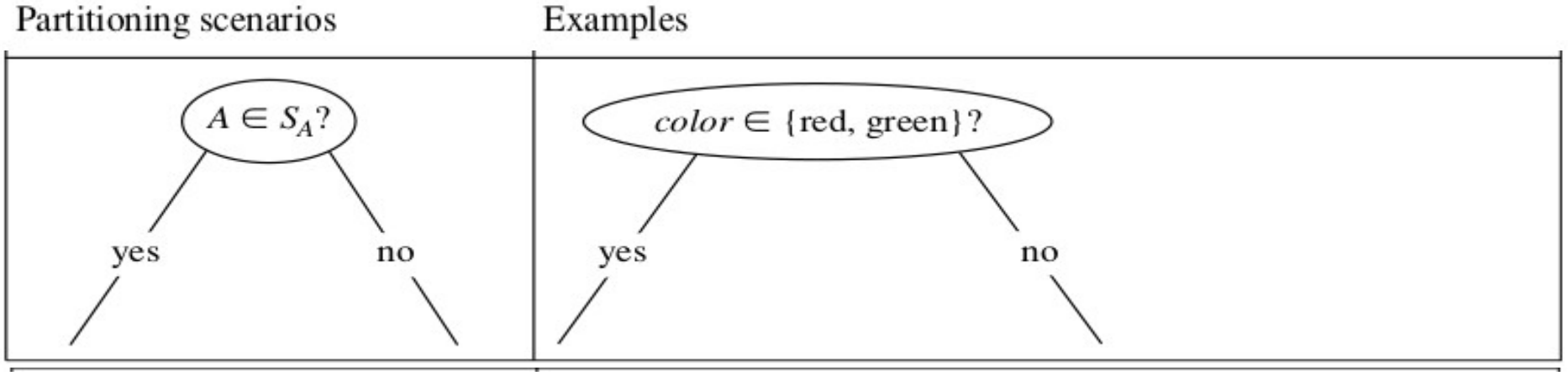
3) A is discrete-valued and a binary tree must be produced: (as dictated by the attribute selection measure or algorithm being used):

The test at node N is of the form "A ∈ S A ?," where S A is the splitting subset for A, returned by Attribute selection method as part of the splitting criterion.

It is a subset of the known values of A. If a given tuple has value a j of A and if aj ∈ S A , then the test at node N is satisfied. Two branches are grown from N.

By convention, the left branch out of N is labelled yes so that D1 corresponds to the subset of class-labelled tuples inD that satisfy the test.

The right branch out of N is labelled no so that D2 corresponds to the subset of class-labelled tuples from D that do not satisfy the test.



Three Termination Conditions

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 res
9) attribute
10) for each ou
// partition the tuples and grow subtrees for each partition
11)
12)
13)
14)
15) return N ;

Termination Condition - 1

All the tuples in partition D (represented at node N) belong to the same class (Step 2 and 3)

Three Termination Conditions

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 res
9) attribute
10) for each ou
// partition the tuples and grow subtrees for each partition
11)
12)
13)
14)
15) return N ;

Termination Condition - 2

No remaining attributes on which the tuples may be further partitioned (step 4).
In this case, majority voting is employed (step 5).
This involves converting node N into a leaf and
labelling it with the most common class in D.
Alternatively, the class distribution of the node tuples may be stored.

Three Termination Conditions

There are no tuples for a given branch, that is, a partition D j is empty (step 12).
In this case, a leaf is created with the majority class in D (step 13)

Termination Condition - 3

```
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 ;
```

Design Issues of DT Induction

1. How should the training records be split?

Each recursive step of the tree-growing process must select an attribute test condition to divide the records into smaller subsets.

To implement this step, the algorithm must provide a method for specifying the test condition for different attribute types as well as an objective measure for evaluating the goodness of each test condition.

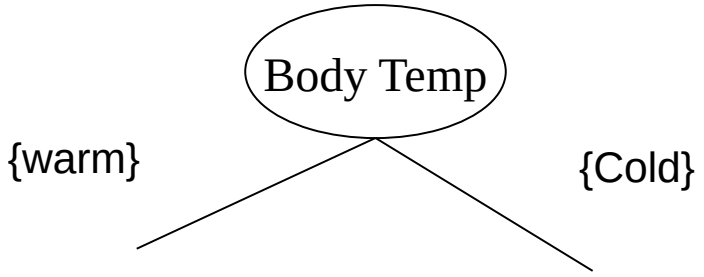
2. How should the splitting procedure stop?

A stopping condition is needed to terminate the tree-growing process. A possible strategy is to continue expanding a node until either all the records belong to the same class or all the records have identical attribute values.

Although both conditions are sufficient to stop any decision tree induction algorithm, other criteria can be imposed to allow the tree-growing procedure to terminate earlier.

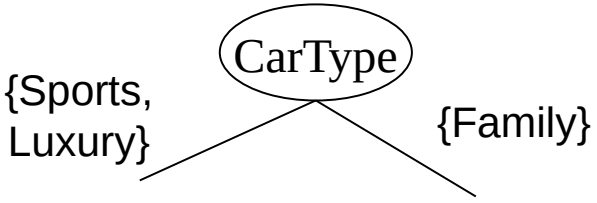
Methods for Expressing Attribute Test Conditions

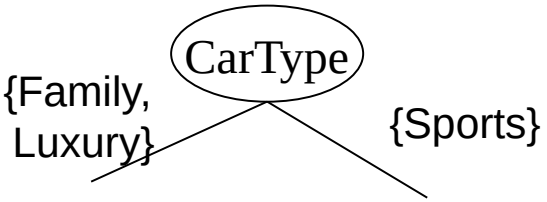
- Binary Attributes:
- Nominal Attributes:
- Ordinal Attributes:
- Continuous Attributes:



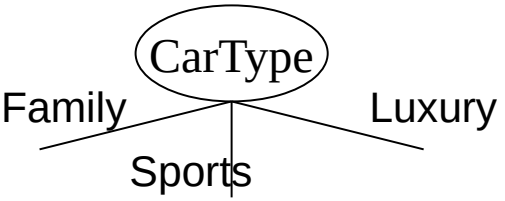
Methods for Expressing Attribute Test Conditions

- Binary Attributes:
- Nominal Attributes:
- Ordinal Attributes:
- Continuous Attributes:





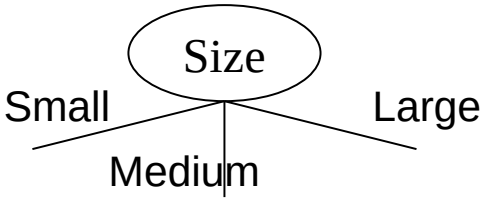
Binary Split [By Grouping attribute values]



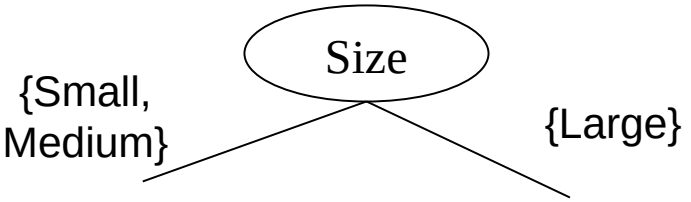
Multiway Split

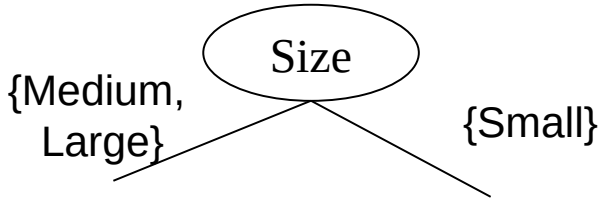
Methods for Expressing Attribute Test Conditions

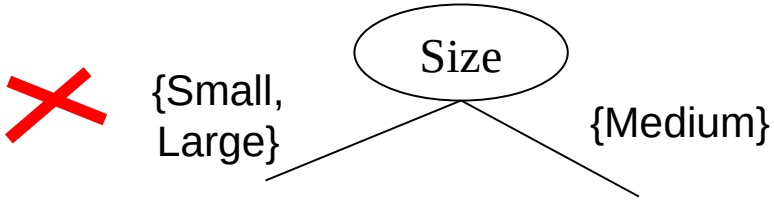
- Binary Attributes:
- Nominal Attributes:
- Ordinal Attributes:
- Continuous Attributes:



Values can be grouped as long as the grouping does not violate the order property of the attribute values.

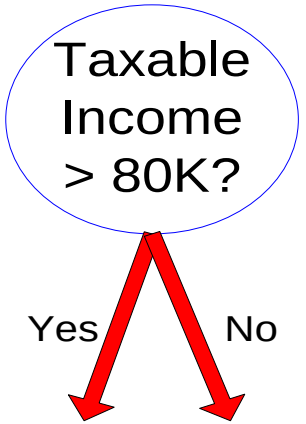




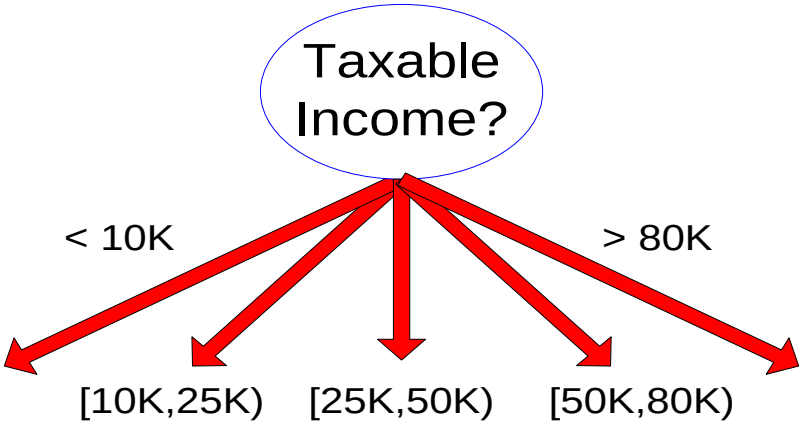


Methods for Expressing Attribute Test Conditions

- Binary Attributes:
- Nominal Attributes:
- Ordinal Attributes:
- Continuous Attributes:



(i) Binary split



(ii) Multi-way split

Attribute Selection Measure: Information Gain: Example-2 : with continuous attributes

1. Let attribute A be a continuous-valued attribute.

2. Standard method: binary splits

3. Must determine the best split point for A

   Sort the value A in increasing order

   Typically, the midpoint is a split point

- $(a_i + a_{i+1})/2$ is the midpoint between the values of a i and a i+1

  Therefore, given v values of A, then v-1 possible splits are evaluated.

The point with the minimum expected information requirement for A is selected as the split- point for A

4. Split: D 1 is the set of instances in D satisfying A ≤ split-point, and D 2 is the set of instances in D satisfying A > split-point

Information Gain: Example-2

5. Split on temperature attribute:

64	65	68	69	70	71	72	75	80	81	83	85
Y	N	Y	Y	Y	N	Y,N	Y,Y	N	Y	Y	N

Information Gain: Example-2

5. Split on temperature attribute:

64	65	68	69	70	71	72	75	80	81	83	85
Y	N	Y	Y	Y	N	Y,N	Y,Y	N	Y	Y	N

E.g.

Let's take split point = **71.5** then distribution of data instances is as below

Temp ≤ 71.5 : **?** Yes, **?** No – D 1

Temp > 71.5 : **?** Yes, **?** No – D 2

Information Gain: Example-2

5. Split on temperature attribute:

64	65	68	69	70	71	72	75	80	81	83	85
Y	N	Y	Y	Y	N	Y,N	Y,Y	N	Y	Y	N

E.g.

Let's take split point = **71.5** then distribution of data instances is as below

Temp ≤ 71.5 : 4 Yes, 2 No – D 1

Temp > 71.5 : 5 Yes, 3 No – D 2

Now, information required at this split point can be calculated as below:

Info

= **?**

Information Gain: Example-2

5. Split on temperature attribute:

64	65	68	69	70	71	72	75	80	81	83	85
Y	N	Y	Y	Y	N	Y,N	Y,Y	N	Y	Y	N

E.g.

Let's take split point = **71.5** then distribution of data instances is as below

Temp ≤ 71.5 : 4 Yes, 2 No – D 1

Temp > 71.5 : 5 Yes, 3 No – D 2

Now, information required at this split point can be calculated as below:

Info

= 6/14 x I(4,2) + 8/14 x I(5,3)

= 6/14 x (-4/6 log 2 4/6 – 2/6 log 2 2/6) + 8/14 x (-5/8 log 2 5/8 – 3/8 log 2 3/8)

= 0.939 bits

6. Similarly find information required at every split point.

7. The point with the **minimum expected information requirement** for A **is selected as the split-point** for A