Classification using Decision Tree

CSE-454: Data Warehousing and Data Mining Sessional

MD. JAKARIA
LECTURER
DEPT. OF CSE, MIST

Classification

Definition: In machine learning, classification refers to a predictive modeling problem where a class label is predicted for a given example of input data.

Examples of classification problems include:

- Given an example, classify if it is spam or not.
- Given a handwritten character, classify it as one of the known characters.
- Given recent user behavior, classify as churn or not.

There are perhaps four main types of classification tasks that you may encounter; they are:

- Binary Classification
- Multi-Class Classification
- Multi-Label Classification
- Imbalanced Classification

Decision Tree Classification

Definition: A ***decision tree*** represents a function that takes as input a vector of attribute values and returns a “decision” — a single output value.

As an example, we will build a decision tree to decide whether to wait for a table at a restaurant. The list of attributes that we will consider:

1. *Alternate*
2. *Bar*
3. *Fri/Sat*
4. *Hungry*
5. *Patrons*
6. *Price*
7. *Raining*
8. *Reservation*
9. *Type*
10. *Wait Estimate*

Note that every variable has a small set of possible values.

Decision Tree Classification

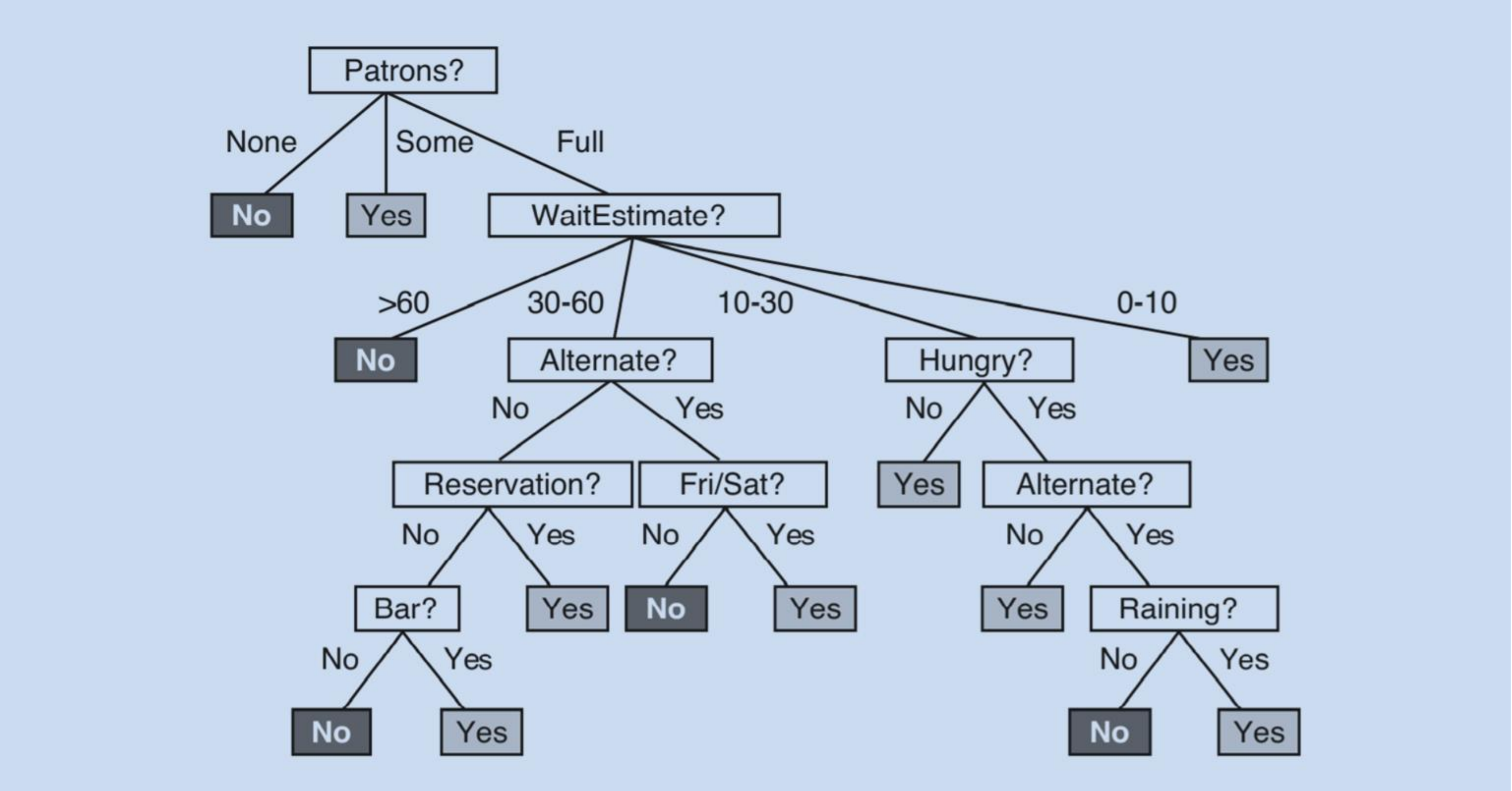


Fig: A decision tree for deciding whether to wait for a table.

Decision Tree Classification

Decision tree -

- A flow-chart-like tree structure
- Internal node denotes a test on an attribute
- Branch represents an outcome of the test
- Leaf nodes represent class labels or class distribution

Use of decision tree:

- Classifying an unknown sample
- Test the attribute values of the sample against the decision tree

Expressiveness of decision trees

A Boolean decision tree is logically equivalent to the assertion that the goal attribute is true if and only if the input attributes satisfy one of the paths leading to a leaf with value true.

$$Goal \Leftrightarrow (Path_1 \vee Path_2 \vee \cdots)$$

where each *Path* is a conjunction of attribute-value tests required to follow that path.

$$Path = (Patrons = Full \wedge WaitEstimate = 0\text{–}10)$$

Constructing decision trees

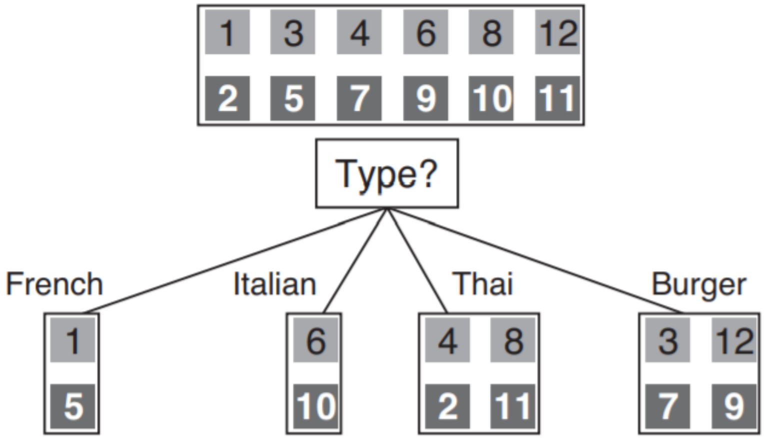
Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*\$\$\$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*\$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*\$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*\$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*\$\$\$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*\$\$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*\$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*\$\$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*\$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*\$\$\$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*\$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*\$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

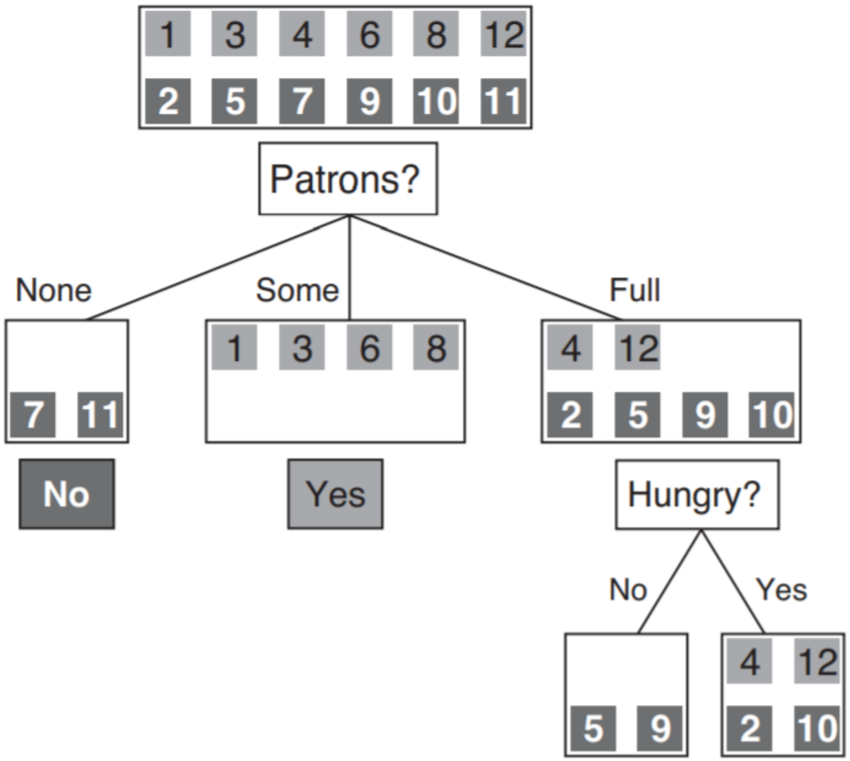
Table: Examples for the restaurant domain.

Constructing decision trees

- We want a tree that is consistent with the examples and is as small as possible
- It is an intractable problem to find the smallest consistent tree; there is no way to efficiently search through the 2^{2^n} trees
- With some simple heuristics, we can find a good approximate solution: a small (but not smallest) consistent tree
- The DECISION-TREE-LEARNING algorithm adopts a greedy divide-and-conquer strategy: always test the most important attribute first.
- By "most important attribute," we mean the one that makes the most difference to the classification of an example.
    - ***Type*** is a poor attribute
    - ***Patrons*** is a fairly important attribute

Constructing decision trees





Constructing decision trees

Basic steps (a greedy approach)

- Tree is constructed in a top-down recursive divide-and-conquer manner
- At start, all the training examples are at the root
- Attributes are categorical (if continuous-valued, they can be discretized in advance)
- Examples are partitioned recursively based on selected attributes.
- Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)

Constructing decision trees

Conditions for stopping partitioning

- All samples for a given node belong to the same class
- There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
- There are no samples left

The DECISION-TREE-LEARNING algorithm

There are four cases to consider

i. If the remaining examples are all positive (or all negative), we can answer *Yes* or *No*.
ii. If there are some positive and some negative examples, then choose the best attribute to split them.
iii. If there are no examples left, we return a default value calculated from the plurality classification of parent node.
iv. If there are no attributes left, but both positive and negative examples, return the plurality classification of the remaining examples.

The DECISION-TREE-LEARNING algorithm

function DECISION-TREE-LEARNING (*examples, attributes, parent_examples*) returns a tree

if *examples* is empty **then return** PLURALITY-VALUE(*parent_examples*)
else if all examples have the same classification **then return** the classification
else if *attributes* is empty **then return** PLURALITY-VALUE(*examples*)
else
  $A \leftarrow \text{argmax}_{a \in \text{attributes}}$ IMPORTANCE(a, *examples*)
  tree ← a new decision tree with root test *A*
  for each value v_k of *A* **do**
    $exs \leftarrow \{ e : e \in examples$ **and** $e.A = v_k \}$
    subtree ← DECISION-TREE-LEARNING(*exs*, *attributes* − *A*, *examples*)
    add a branch to *tree* with label ($A = v_k$) and subtree *subtree*
  return *tree*

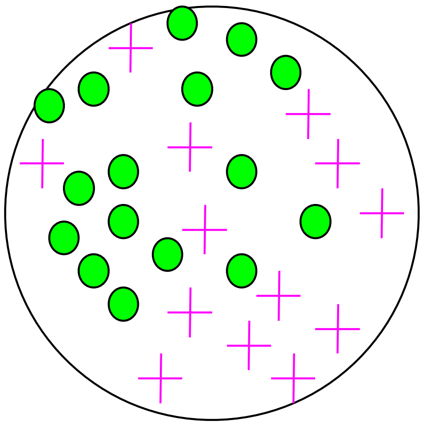
Choosing attribute tests

- The goal is to have the resulting decision tree as small as possible (Occam's Razor)
  - But, finding the minimal decision tree consistent with the data is NP-hard
- The recursive algorithm is a greedy heuristic search for a simple tree, but cannot guarantee optimality.
- The main decision in the algorithm is the selection of the next attribute to condition on.

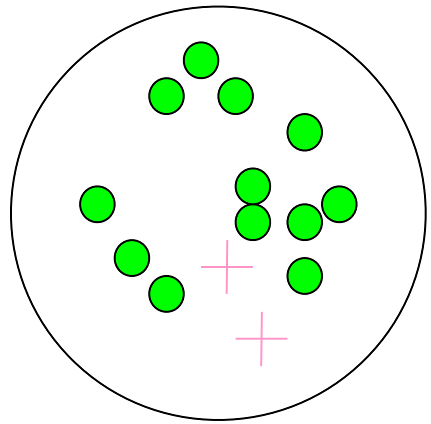
- Which is the best attribute?
  — The one which will result in the smallest tree
  — Heuristic: choose the attribute that produces the "purest" nodes
- Need a good measure of purity!
  — Maximal when?
  — Minimal when?

Impurity/Entropy (informal)

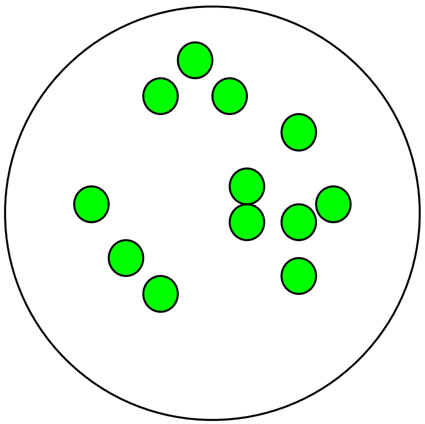
Measures the level of impurity in a group of examples



Very Impure Group



Less Impure Group



Minimum Impure Group

Impurity/Entropy (informal)

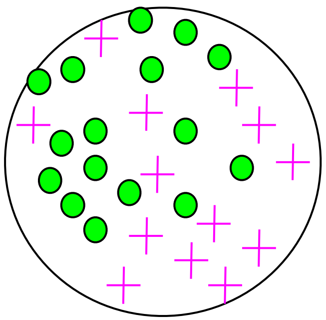
Measures the level of impurity in a group of examples

$$Entropy = -\sum_i p_i log_2 p_i$$

- Here, p_i is the probability of class i.
- Compute it as the proportion of class i in the set.

$$Entropy = -(\frac{16}{30} log_2 \frac{16}{30} + \frac{14}{30} log_2 \frac{14}{30})$$

$$= 0.996$$



Impurity/Entropy (informal)

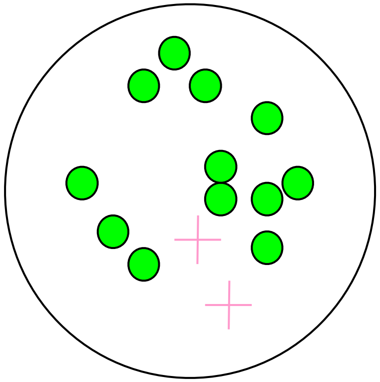
Measures the level of impurity in a group of examples

$$Entropy = -\sum_{i} p_i log_2 p_i$$

- Here, p_i is the probability of class i.
- Compute it as the proportion of class i in the set.

$$Entropy = -(\frac{12}{14} log_2 \frac{12}{14} + \frac{2}{14} log_2 \frac{2}{14})$$

$$= 0.591$$



Impurity/Entropy (informal)

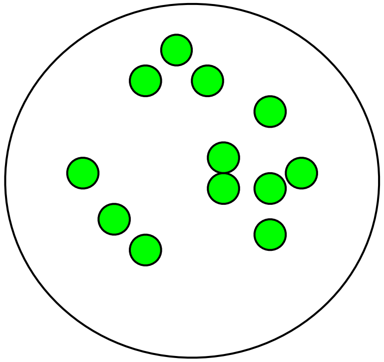
Measures the level of impurity in a group of examples

$$Entropy = -\sum_i p_i log_2 p_i$$

- Here, p_i is the probability of class i.
- Compute it as the proportion of class i in the set.

$$Entropy = -(\frac{12}{12} log_2 \frac{12}{12} + \frac{0}{12} log_2 \frac{0}{12})$$

$$= 0$$



Information Gain

- We want to determine which attribute in a given set of training feature vectors is most useful for discriminating between the classes to be learned.

- Information gain tells us how important a given attribute of the feature vectors is.

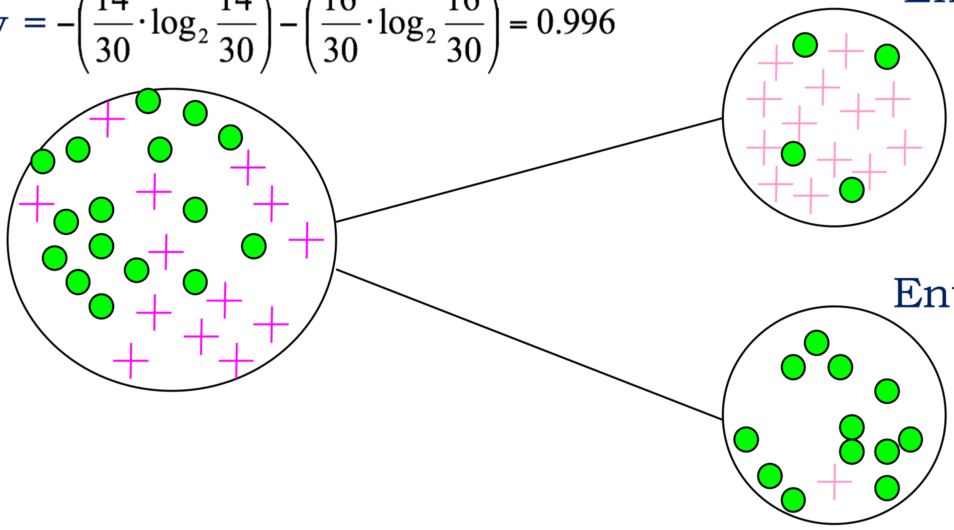
- We will use it to decide the ordering of attributes in the nodes of a decision tree.

$$Information\ Gain = Entropy(parent) - [Average\ Entropy(children)]$$

Information Gain

Entropy $= -\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right) - \left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right) = 0.996$

Entropy $= -\left(\frac{13}{17}\cdot\log_2\frac{13}{17}\right) - \left(\frac{4}{17}\cdot\log_2\frac{4}{17}\right) = 0.787$



Entropy $= -\left(\frac{1}{13}\cdot\log_2\frac{1}{13}\right) - \left(\frac{12}{13}\cdot\log_2\frac{12}{13}\right) = 0.391$

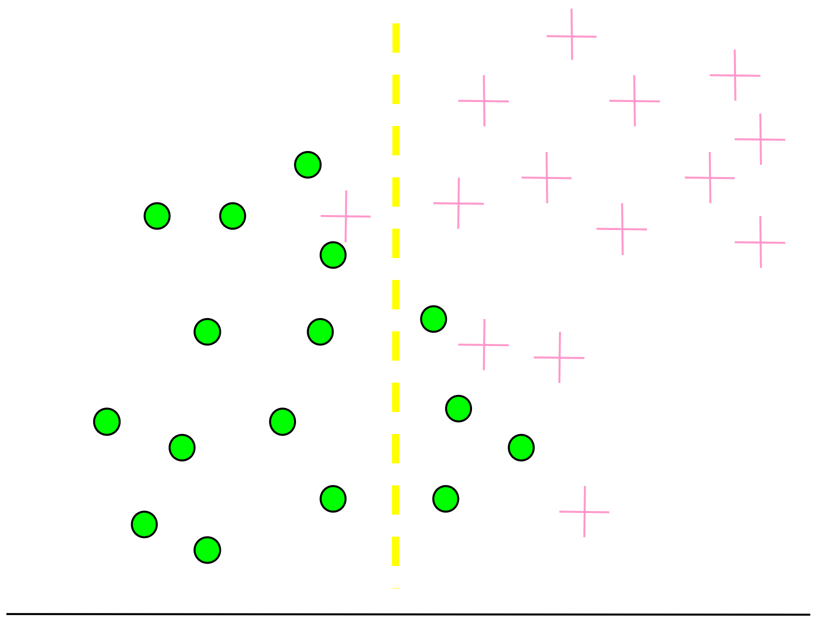
(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right) + \left(\frac{13}{30}\cdot 0.391\right) = 0.615$

$$Information\ Gain = Entropy(parent) - [Average\ Entropy(children)]$$

$= 0.996 - 0.615 = 0.38$

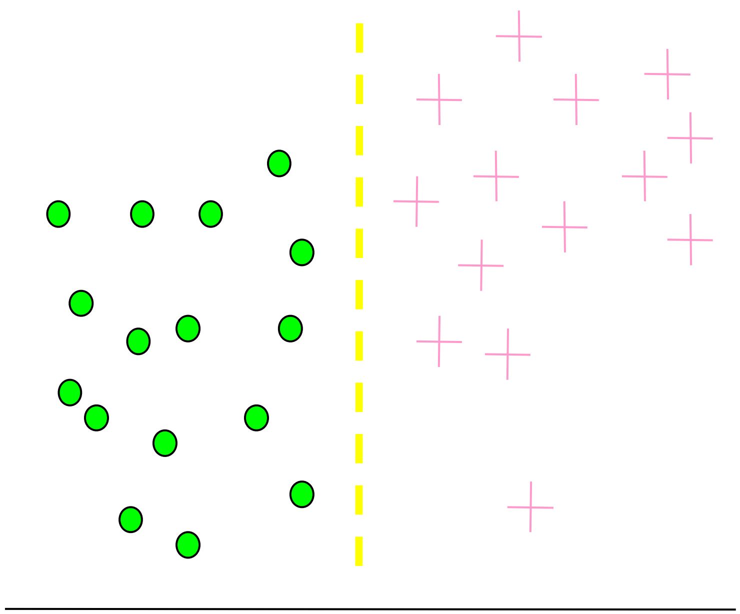
Information Gain

Which test is more informative?



Less or equal 50K   Over 50k

Split over whether Balance exceeds 50K



Unemployed   Employed

Split over whether applicant is employed

The DECISION-TREE-LEARNING algorithm

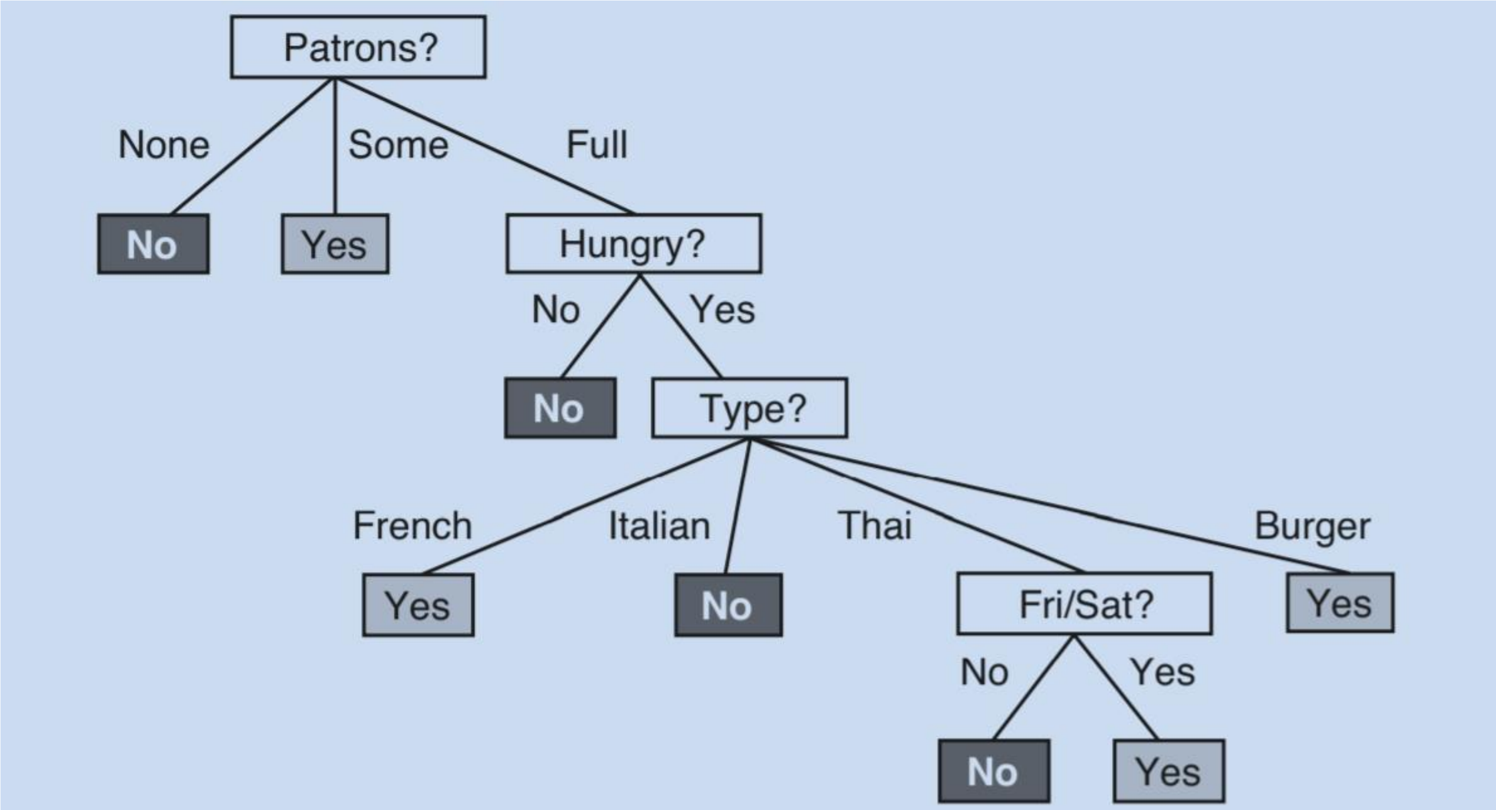


Fig: The decision tree induced from the 12-example training set.

References:

i. Artificial Intelligence: A Modern Approach by Peter Norvig and Stuart J. Russell
Chapter 18 (18.3 Learning Decision Tree)