Classification

- Classification is the task of assigning objects to one of several predefined categories.
- It is an important problem in many applications
  - Detecting spam email messages based on the message header and content.
  - Categorizing cells as malignant or benign based on the results of MRI scans.
  - Classifying galaxies based on their shapes.

Classification

- The input data for a classification task is a collection of records.
- Each record, also known as an instance or example, is characterized by a tuple (**x**, y)
  - **x** is the attribute set
  - y is the class label, also known as category or target attribute.
- The class label is a discrete attribute.

Classification

- Classification is the task of learning a target function f that maps each attribute set **x** to one of the predefined class labels y.
- The target function is also known as a classification model.
- A classification model is useful for the following purposes
  - Descriptive modeling
  - Predictive modeling

Classification

- A classification technique (or classifier) is a systematic approach to perform classification on an input data set.
- Examples include
  - Decision tree classifiers
  - Neural networks
  - Support vector machines

Classification

- A classification technique employs a learning algorithm to identify a model that best fits the relationship between the attribute set and the class label of the input data.
- The model generated by a learning algorithm should
  - Fit the input data well and
  - Correctly predict the class labels of records it has never seen before.
- A key objective of the learning algorithm is to build models with good generalization capability.

Classification

- First, a training set consisting of records whose class labels are known must be provided.
- The training set is used to build a classification model.
- This model is subsequently applied to the test set, which consists of records which are different from those in the training set.

Confusion matrix

- Evaluation of the performance of the model is based on the counts of correctly and incorrectly predicted test records.
- These counts are tabulated in a table known as a confusion matrix.
- Each entry a_{ij} in this table denotes the number of records from class i predicted to be of class j.

Confusion matrix

		Predicted Class	
		Class=1	Class=0
Actual Class	Class=1	a_{11}	a_{10}
	Class=0	a_{01}	a_{00}

Confusion matrix

- The total number of correct predictions made by the model is $a_{11}+a_{00}$.
- The total number of incorrect predictions is $a_{10}+a_{01}$.

Confusion matrix

- The information in a confusion matrix can be summarized with the following two measures
  - Accuracy

$$Accuracy = \frac{a_{11} + a_{00}}{a_{11} + a_{10} + a_{01} + a_{00}}$$

  - Error rate

$$Error\ Rate = \frac{a_{10} + a_{01}}{a_{11} + a_{10} + a_{01} + a_{00}}$$

- Most classification algorithms aim at attaining the highest accuracy, or equivalently, the lowest error rate when applied to the test set.

Decision tree

- We can solve a classification problem by asking a series of carefully crafted questions about the attributes of the test record.
- Each time we receive an answer, a follow-up question is asked.
- This process is continued until we reach a conclusion about the class label of the record.

Decision tree

- The series of questions and answers can be organized in the form of a decision tree.
- It is a hierarchical structure consisting of nodes and directed edges.
- The tree has three types of nodes
  - A root node that has no incoming edges.
  - Internal nodes, each of which has exactly one incoming edge and a number of outgoing edges.
  - Leaf or terminal nodes, each of which has exactly one incoming edge and no outgoing edges.

Decision tree

- In a decision tree, each leaf node is assigned a class label.
- The non-terminal nodes, which include the root and other internal nodes, contain attribute test conditions to separate records that have different characteristics.

Decision tree

- Classifying a test record is straightforward once a decision tree has been constructed.
- Starting from the root node, we apply the test condition.
- We then follow the appropriate branch based on the outcome of the test.
- This will lead us either to
  - Another internal node, at which a new test condition is applied, or
  - A leaf node.
- The class label associated with the leaf node is then assigned to the record.

Decision tree construction

- Efficient algorithms have been developed to induce a reasonably accurate, although suboptimal, decision tree in a reasonable amount of time.
- These algorithms usually employ a greedy strategy that makes a series of locally optimal decisions about which attribute to use for partitioning the data.

Decision tree construction

- A decision tree is grown in a recursive fashion by partitioning the training records into successively purer subsets.
- We suppose
  - U_s is the set of training records that are associated with node s.
  - $C=\{c_1, c_2, \ldots\ldots c_K\}$ is the set of class labels.

Decision tree construction

- If all the records in U_s belong to the same class c_k, then s is a leaf node labeled as c_k.
- If U_s contains records that belong to more than one class,
  - An attribute test condition is selected to partition the records into smaller subsets.
  - A child node is created for each outcome of the test condition.
  - The records in U_s are distributed to the children based on the outcomes.
- The algorithm is then recursively applied to each child node.

Decision tree construction

- For each node, let $p(c_k)$ denotes the fraction of training records from class c_k.
- In most cases, the leaf node is assigned to the class that has the majority number of training records.
- The fraction $p(c_k)$ for a node can also be used to estimate the probability that a record assigned to that node belongs to class k.

Decision tree construction

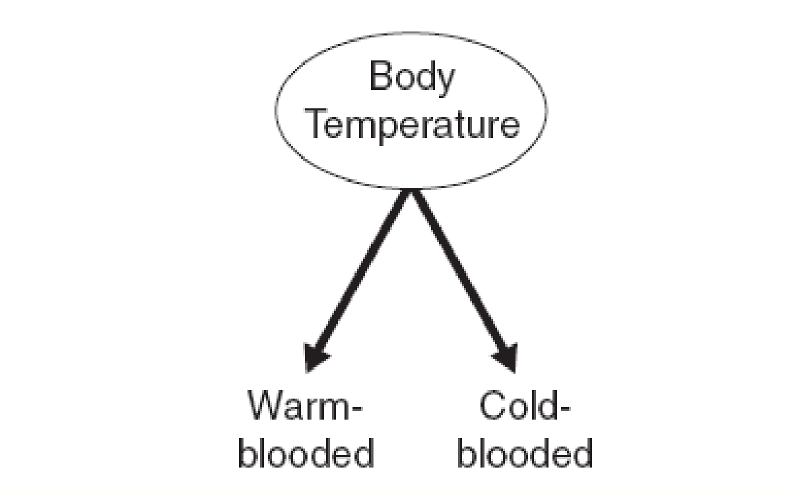
- Decision trees that are too large are susceptible to a phenomenon known as overfitting.
- A tree pruning step can be performed to reduce the size of the decision tree.
- Pruning helps by trimming the tree branches in a way that improves the generalization error.

Attribute test

- 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 and
  - An objective measure for evaluating the goodness of each test condition.

Attribute test

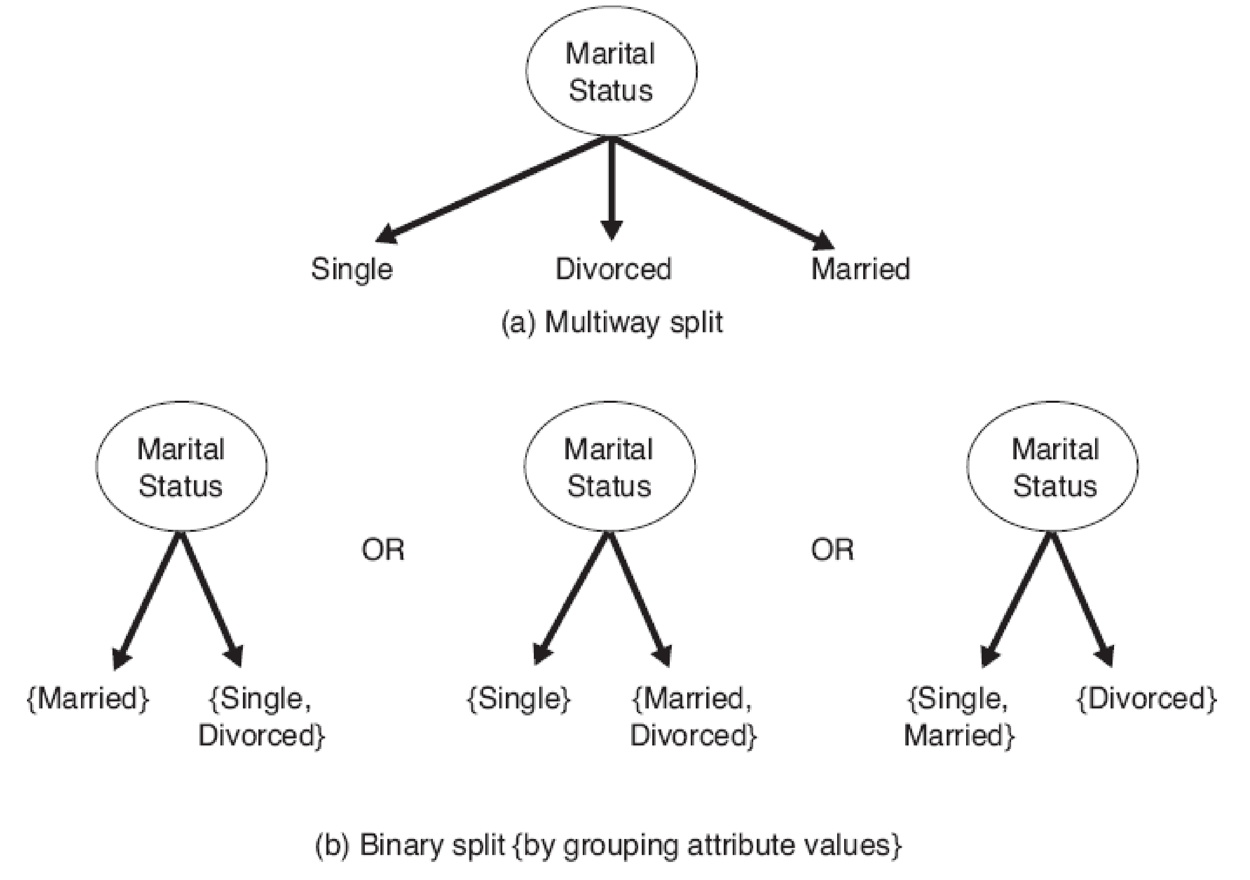
- Binary attributes
  - The test condition for a binary attribute generates two possible outcomes.



Attribute test

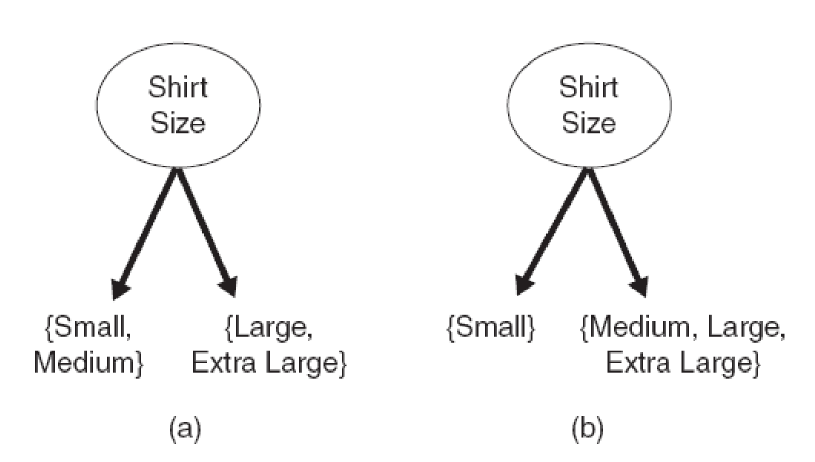
- Nominal attributes
  - A nominal attribute can produce binary or multi-way splits.
  - There are $2^{S-1}-1$ ways of creating a binary partition of S attribute values.
  - For a multi-way split, the number of outcomes depends on the number of distinct values for the corresponding attribute.

Attribute test



Attribute test

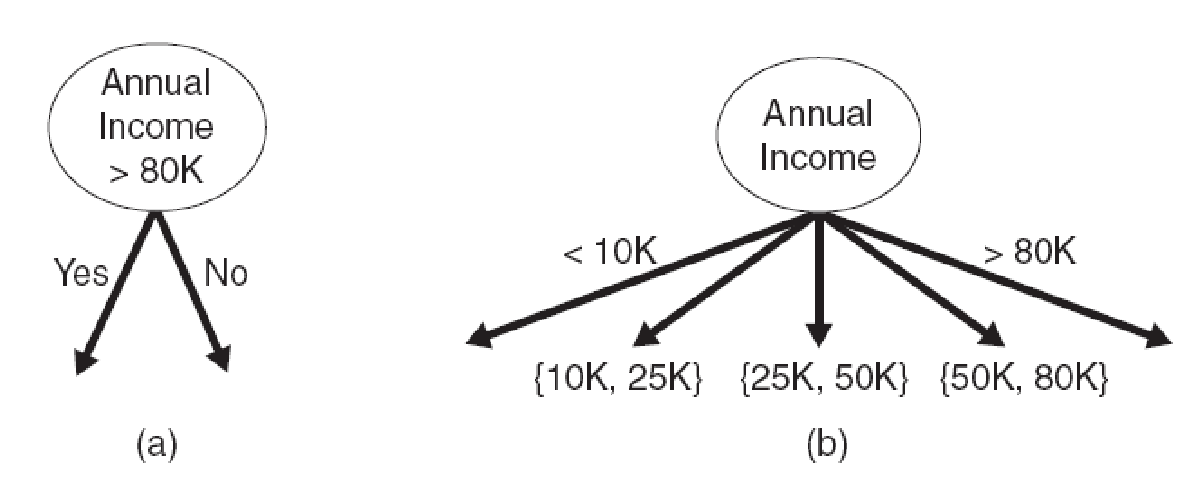
- Ordinal attributes
  - Ordinal attributes can also produce binary or multi-way splits.
  - Ordinal attributes can be grouped as long as the grouping does not violate the order property of the attribute values.



Attribute test

- Continuous attributes
  - The test condition can be expressed as a comparison test $x \le T$ or $x > T$.
  - For the binary case
    - The decision tree algorithm must consider all possible split positions T, and
    - Select the one that produces the best partition.
  - For the multi-way split,
    - The algorithm must consider multiple split positions.

Attribute test



Decision tree construction

- We consider the problem of using a decision tree to predict the number of participants in a marathon race requiring medical attention.
- This number depends on attributes such as
  - Temperature forecast (**TEMP**)
  - Humidity forecast (**HUMID**)
  - Air pollution forecast (**AIR**)

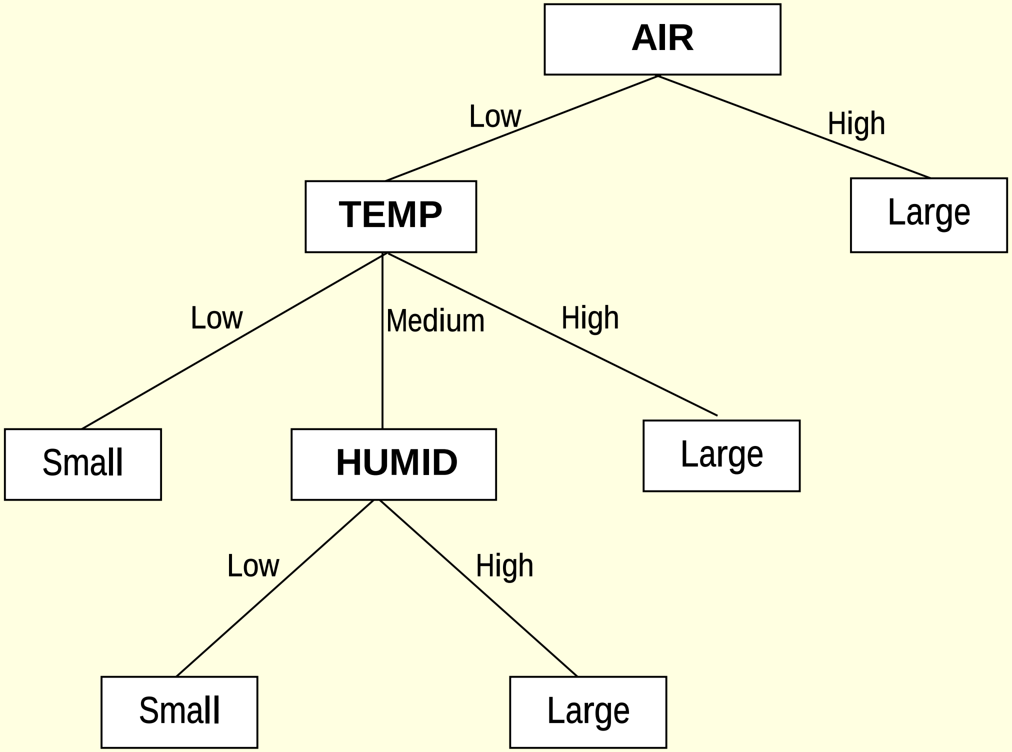
Decision tree construction

Condition	**TEMP**	**HUMID**	**AIR**	Number of marathon participants requiring medical attention
1	High	High	Low	Large
2	High	High	High	Large
3	Medium	High	Low	Large
4	Low	Low	Low	Small
5	Low	Low	High	Large
6	Medium	Low	Low	Small
7	Medium	Low	High	Large
8	Medium	High	High	Large
9	High	Low	Low	Large
10	High	Low	High	Large

Decision tree construction

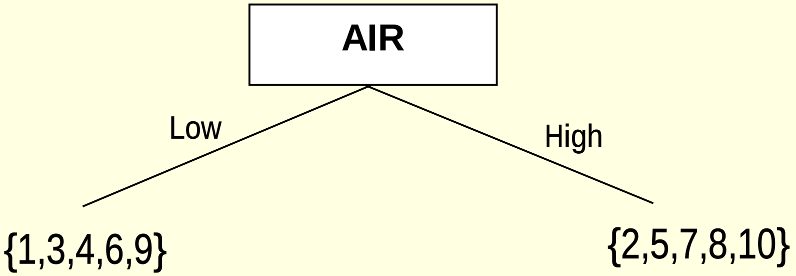
- In a decision tree, each internal node represents a particular attribute, e.g., **TEMP** or **AIR**.
- Each possible value of that attribute corresponds to a branch of the tree.
- Leaf nodes represent classifications, such as Large or Small number of participants requiring medical attention.

Decision tree construction



Decision tree construction

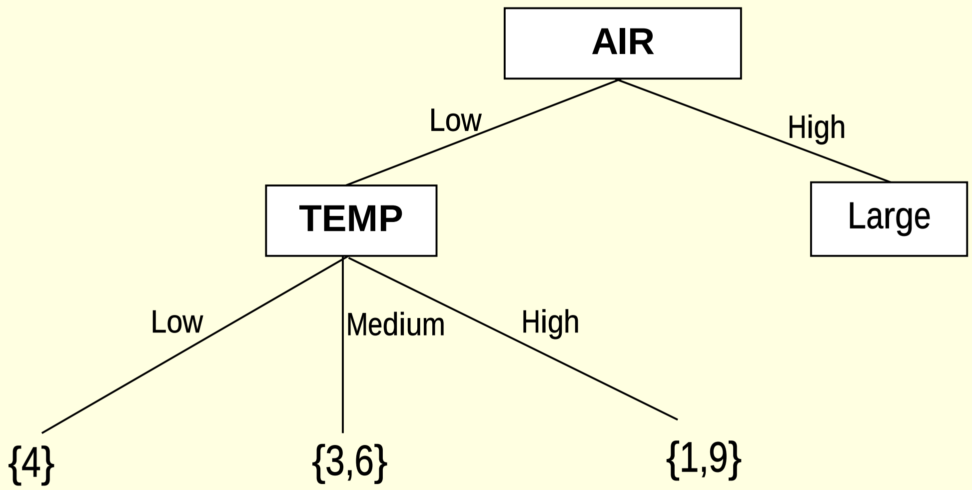
- Suppose **AIR** is selected as the first attribute.
- This partitions the examples as follows.



Decision tree construction

- Since the entries of the set {2,5,7,8,10} all correspond to the case of a large number of participants requiring medical attention, a leaf node is formed.
- On the other hand, for the set {1,3,4,6,9}
  - **TEMP** is selected as the next attribute to be tested.
  - This further divides this partition into {4}, {3,6} and {1,9}.

Decision tree construction



Information theory

- Each attribute reduces a certain amount of uncertainty in the classification process.
- We calculate the amount of uncertainty reduced by the selection of each attribute.
- We then select the attribute that provides the greatest uncertainty reduction.

Information theory

- Information theory provides a mathematical formulation for measuring how much information a message contains.
- We consider the case where a message is selected among a set of possible messages and transmitted.
- The information content of a message depends on
  - The size of this message set, and
  - The frequency with which each possible message occurs.

Information theory

- The amount of information in a message with occurrence probability p is defined as $-\log_2 p$.
- Suppose we are given
  - a set of messages, $C=\{c_1,c_2,\ldots\ldots,c_K\}$
  - the occurrence probability $p(c_k)$ of each c_k.
- We define the entropy I as the expected information content of a message in C :

$$I = -\sum_{k=1}^{K} p(c_k)\log_2 p(c_k)$$

- The entropy is measured in bits.

Attribute selection

- We can calculate the entropy of a set of training examples from the occurrence probabilities of the different classes.
- In our example
  - p(Small)=2/10
  - p(Large)=8/10

Attribute selection

- The set of training instances is denoted as U
- We can calculate the entropy as follows:

$$
\begin{aligned}
I(U) &= -\frac{2}{10}\log_2\left(\frac{2}{10}\right) - \frac{8}{10}\log_2\left(\frac{8}{10}\right) \\
&= -\frac{2}{10}(-2.322) - \frac{8}{10}(-0.322) \\
&= 0.722 \text{ bit}
\end{aligned}
$$

Attribute selection

- The information gain provided by an attribute is the difference between
  1. The degree of uncertainty before including the attribute.
  2. The degree of uncertainty after including the attribute.
- Item 2 above is defined as the weighted average of the entropy values of the child nodes of the attribute.

Attribute selection

- If we select attribute P, with S values, this will partition U into the subsets $\{U_1, U_2, \ldots, U_S\}$.
- The average degree of uncertainty after selecting P is

$$\bar{I}(P) = \sum_{s=1}^{S} \frac{|U_s|}{|U|} I(U_s)$$

Attribute selection

- The information gain associated with attribute P is computed as follows.
  - $gain(P) = I(U) - \bar{I}(P)$
- If the attribute **AIR** is chosen, the examples are partitioned as follows:
  - U_1={1,3,4,6,9}
  - U_2={2,5,7,8,10}

Attribute selection

- The resulting entropy value is

$$\bar{I}(\textbf{AIR}) = \frac{5}{10} I(U_1) + \frac{5}{10} I(U_2)$$

$$= \frac{5}{10}\left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) + \frac{5}{10}(0)$$

$$= 0.485\,\text{bit}$$

Attribute selection

- The information gain can be computed as follows:

$$
\begin{aligned}
gain(\textbf{AIR}) &= I(U) - \bar{I}(\textbf{AIR}) \\
&= 0.722 - 0.485 \\
&= 0.237\,\text{bit}
\end{aligned}
$$

Attribute selection

- For the attribute **TEMP** which partitions the examples into U_1={4,5}, U_2={3,6,7,8} and U_3={1,2,9,10}:

$$\bar{I}(\textbf{TEMP}) = \frac{2}{10} I(U_1) + \frac{4}{10} I(U_2) + \frac{4}{10} I(U_3)$$
$$= \frac{2}{10}(-\frac{1}{2}\log_2 \frac{1}{2} - \frac{1}{2}\log_2 \frac{1}{2}) + \frac{4}{10}(-\frac{1}{4}\log_2 \frac{1}{4} - \frac{3}{4}\log_2 \frac{3}{4}) + \frac{4}{10}(0)$$
$$= 0.525\,\text{bit}$$

$$gain(\textbf{TEMP}) = I(U) - \bar{I}(\textbf{TEMP})$$
$$= 0.722 - 0.525$$
$$= 0.197\,\text{bit}$$

Attribute selection

- For the attribute **HUMID** which partitions the examples into U_1={4,5,6,7,9,10} and U_2={1,2,3,8}:

$$
\begin{aligned}
\bar{I}(\textbf{HUMID}) &= \frac{6}{10} I(U_1) + \frac{4}{10} I(U_2) \\
&= \frac{6}{10}\left(-\frac{2}{6}\log_2\frac{2}{6} - \frac{4}{6}\log_2\frac{4}{6}\right) + \frac{4}{10}(0) \\
&= 0.551\,\text{bit}
\end{aligned}
$$

$$
\begin{aligned}
gain(\textbf{HUMID}) &= I(U) - \bar{I}(\textbf{HUMID}) \\
&= 0.722 - 0.551 \\
&= 0.171\,\text{bit}
\end{aligned}
$$

Attribute selection

- The attribute **AIR** corresponds to the highest information gain.
- As a result, this attribute will be selected.

Continuous attributes

- If attribute P is continuous with value x, we can apply a binary test.
- The outcome of the test depends on a threshold value T.
- There are two possible outcomes:
  - $x \leq T$
  - $x > T$
- The training set is then partitioned into 2 subsets U_1 and U_2.

Continuous attributes

- We apply sorting to values of attribute P to obtain the sequence $\{x_{(1)},x_{(2)},\ldots\ldots,x_{(m)}\}$.
- Any threshold between $x_{(r)}$ and $x_{(r+1)}$ will divide the set into two subsets
  - $\{x_{(1)},x_{(2)},\ldots\ldots,x_{(r)}\}$
  - $\{x_{(r+1)},x_{(r+2)},\ldots\ldots,x_{(m)}\}$
- There are at most m-1 possible splits.

Continuous attributes

- For r=1,......,m-1 such that $x_{(r)} \neq x_{(r+1)}$, the corresponding threshold is chosen as $T_r=(x_{(r)}+x_{(r+1)})/2$.
- We can then calculate the information gain for each T_r
  - $gain(P,T_r) = I(U) - \bar{I}(P,T_r)$

  where $\bar{I}(P,T_r)$ is a function of T_r.
- The threshold T_r which maximizes gain(P,T_r) is then chosen.

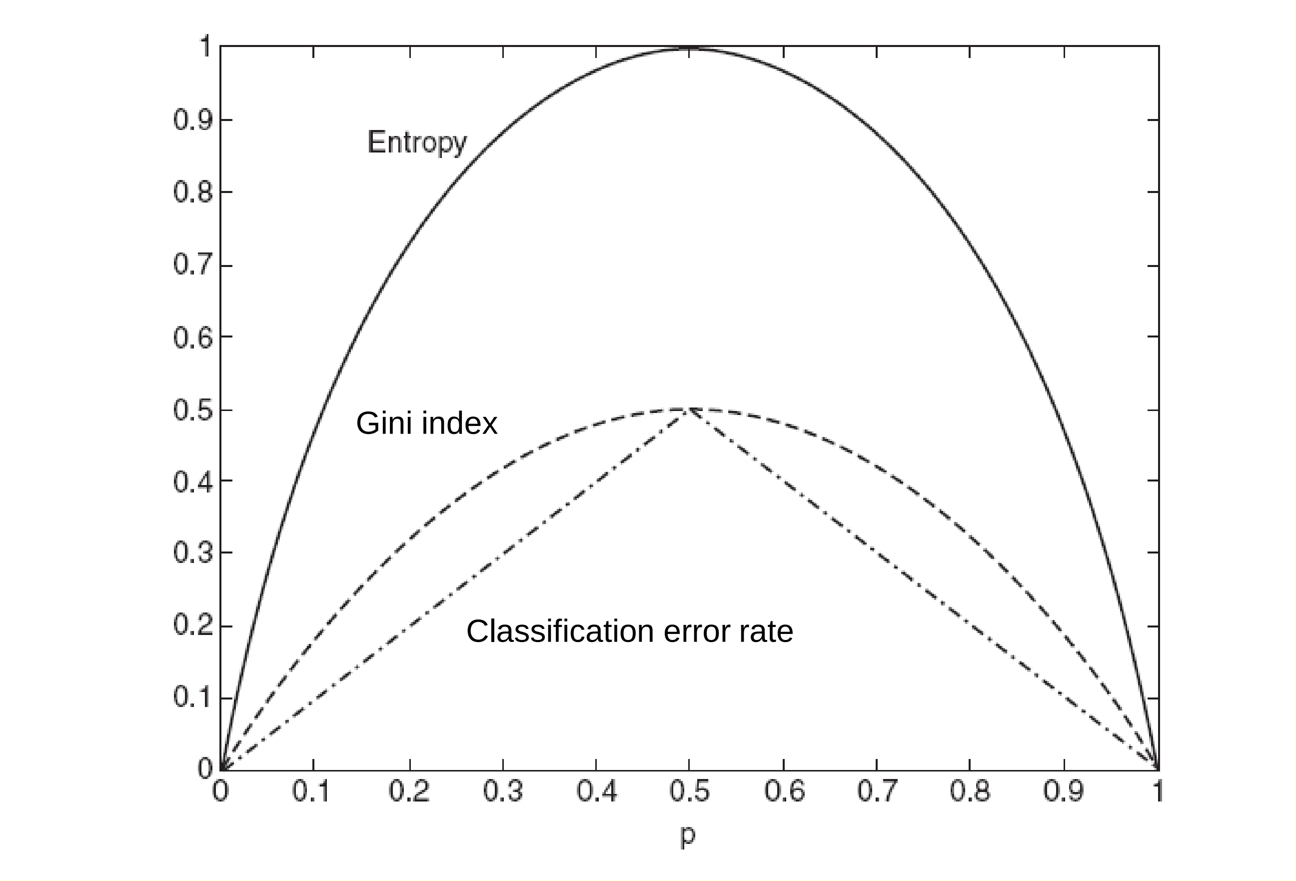
Impurity measures

- The measures developed for selecting the best split are often based on the degree of impurity of the child nodes.
- Besides entropy, other examples of impurity measures include
  - Gini index
    - $$G = 1 - \sum_{k=1}^{K} p(c_k)^2$$
  - Classification error rate
    - $$E = 1 - \max_{k} p(c_k)$$

Impurity measures

- In the following figure, we compare the values of the impurity measures for binary classification problems.
- p refers to the fraction of records that belong to one of the two classes.
- All three measures attain their maximum value when p=0.5.
- The minimum values of the measures are attained when p equals 0 or 1.

Impurity measures



Gain ratio

- Impurity measures such as entropy and Gini index tend to favor attributes that have a large number of possible values.
- In many cases, a test condition that results in a large number of outcomes may not be desirable.
- This is because the number of records associated with each partition is too small to enable us to make any reliable predictions.

Gain ratio

- To solve this problem, we can modify the splitting criterion to take into account the number of possible attribute values.
- In the case of information gain, we can use the gain ratio which is defined as follows

$$Gain\ Ratio = \frac{Gain(P)}{Split\ Info}$$

where

$$Split\ Info = -\sum_{s=1}^{S} \frac{|U_s|}{|U|} \log_2 \frac{|U_s|}{|U|}$$

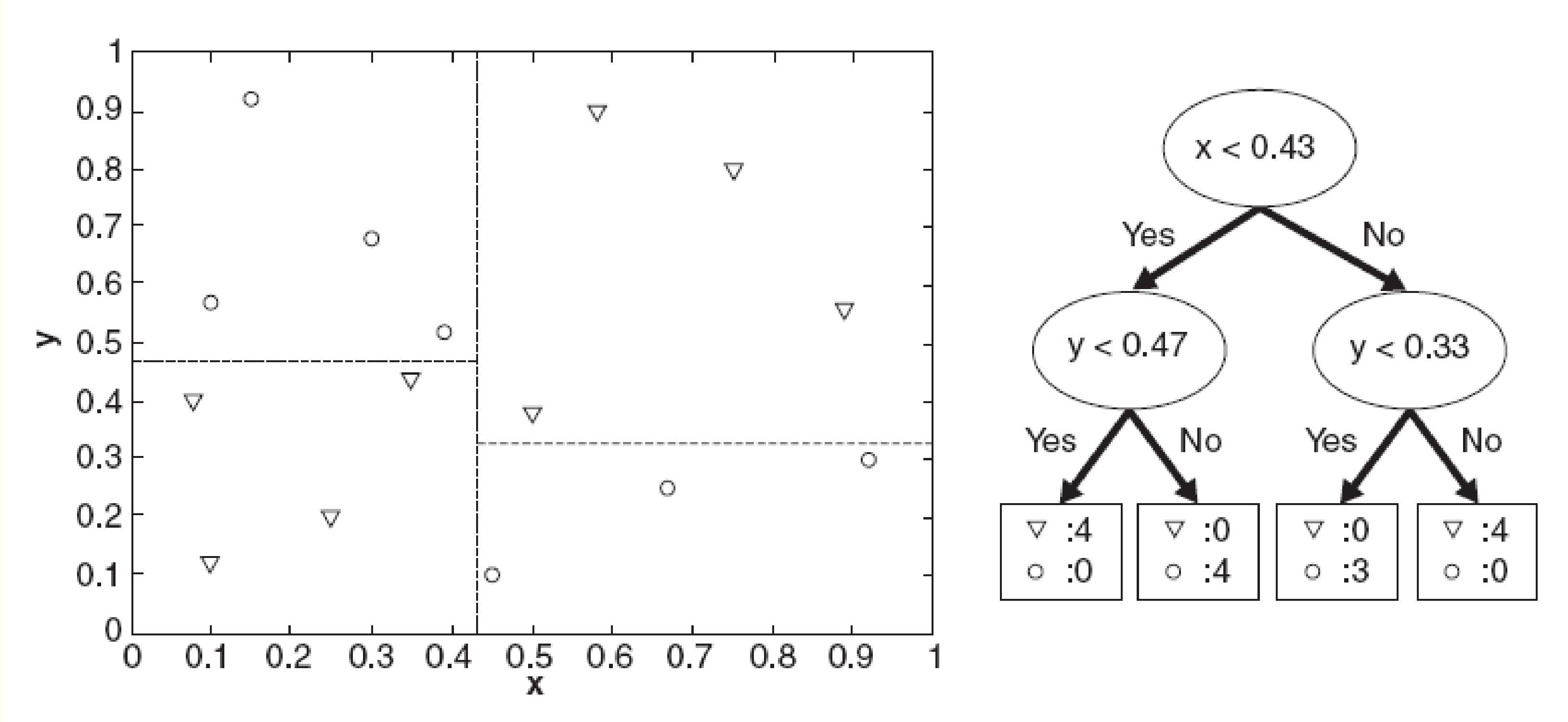
Oblique decision tree

- The test condition described so far involve using only a single attribute at a time.
- The tree-growing procedure can be viewed as the process of partitioning the attribute space into disjoint regions.
- The border between two neighboring regions of different classes is known as a decision boundary.

Oblique decision tree

- Since the test condition involves only a single attribute, the decision boundaries are parallel to the coordinate axes.
- This limits the expressiveness of the decision tree representation for modeling complex relationships among continuous attributes.

Oblique decision tree



Oblique decision tree

- An oblique decision tree allows test conditions that involve more than one attribute.
- The following figure illustrates a data set that cannot be classified effectively by a conventional decision tree.
- This data set can be easily represented by a single node of an oblique decision tree with the test condition $x+y<1$
- However, finding the optimal test condition for a given node can be computationally expensive.

Oblique decision tree