Data Mining

Classification, Clustering & Association

Agenda

1. Introduction to Data Mining
   - What is Data Mining,

1. Classification
   - Decision tree
   - Naïve Bayes Classifier

1. Clustering
   - K-means Clustering
   - Agglomerative

1. Frequent Pattern Mining
   - Apriori
   - FP Growth

Motivation

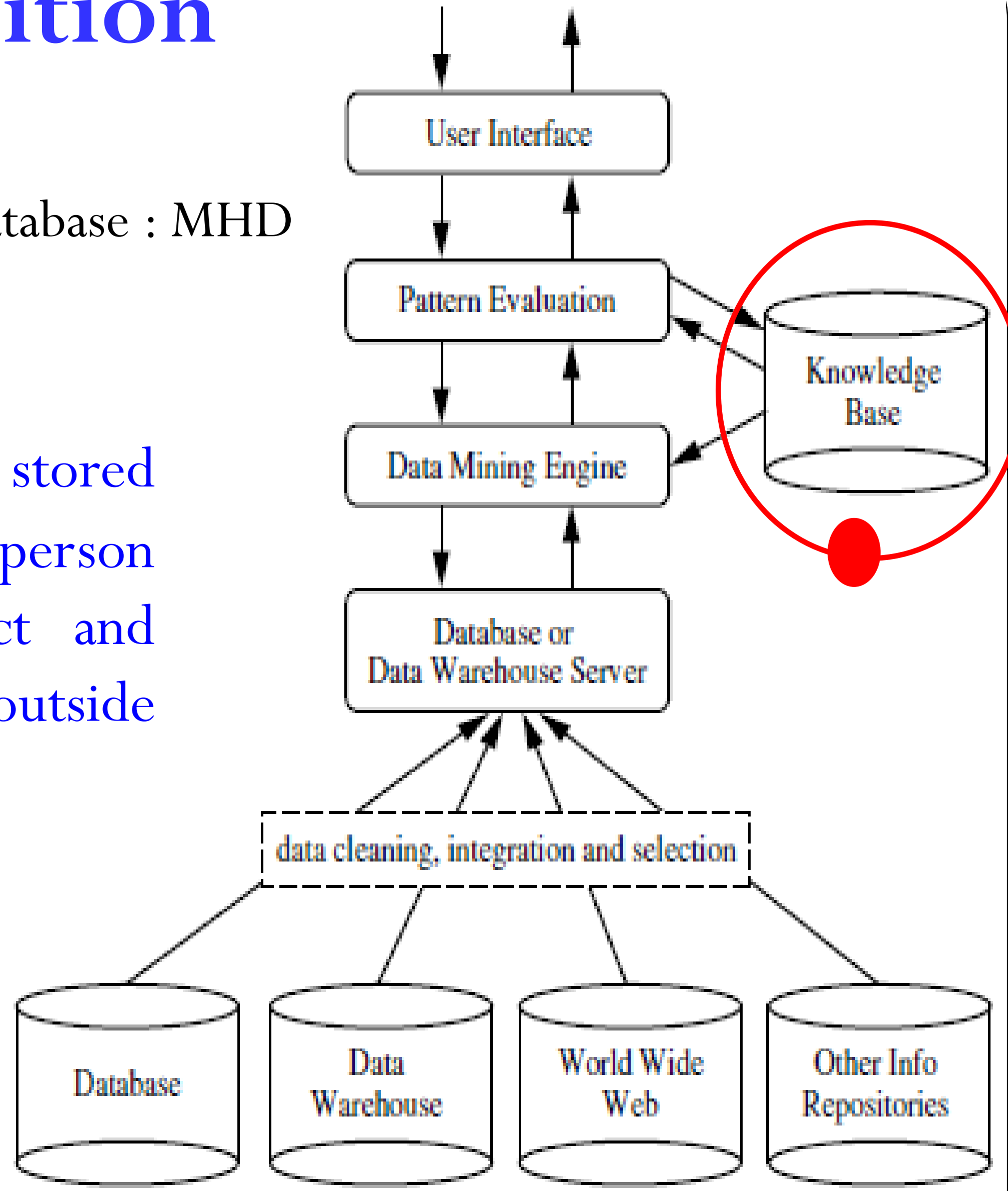
- Data explosion problem
  - Automated data collection tools and mature database technology lead to tremendous amounts of data stored in databases, data warehouses and other information repositories

Need to convert such data into Information and **Knowledge**.

DM Definition

- Finding **Hidden Information** in Database : MHD

Knowledge refers to stored information or models used by a person or machine to interpret, predict and appropriately respond to the outside world. **(Haykin)**



Data explosion problem : Need to convert such data into Information and **Knowledge**.

DM Definition

- Process of semi-automatically analyzing large databases to find patterns that are:
  - valid: hold on new data with some certainity
  - novel: non-obvious to the system
  - useful: should be possible to act on the item
  - understandable: humans should be able to interpret the pattern
- Finding Hidden Information in Database (MHD).
- Also known as Knowledge Discovery in Databases (KDD)

DM Definition

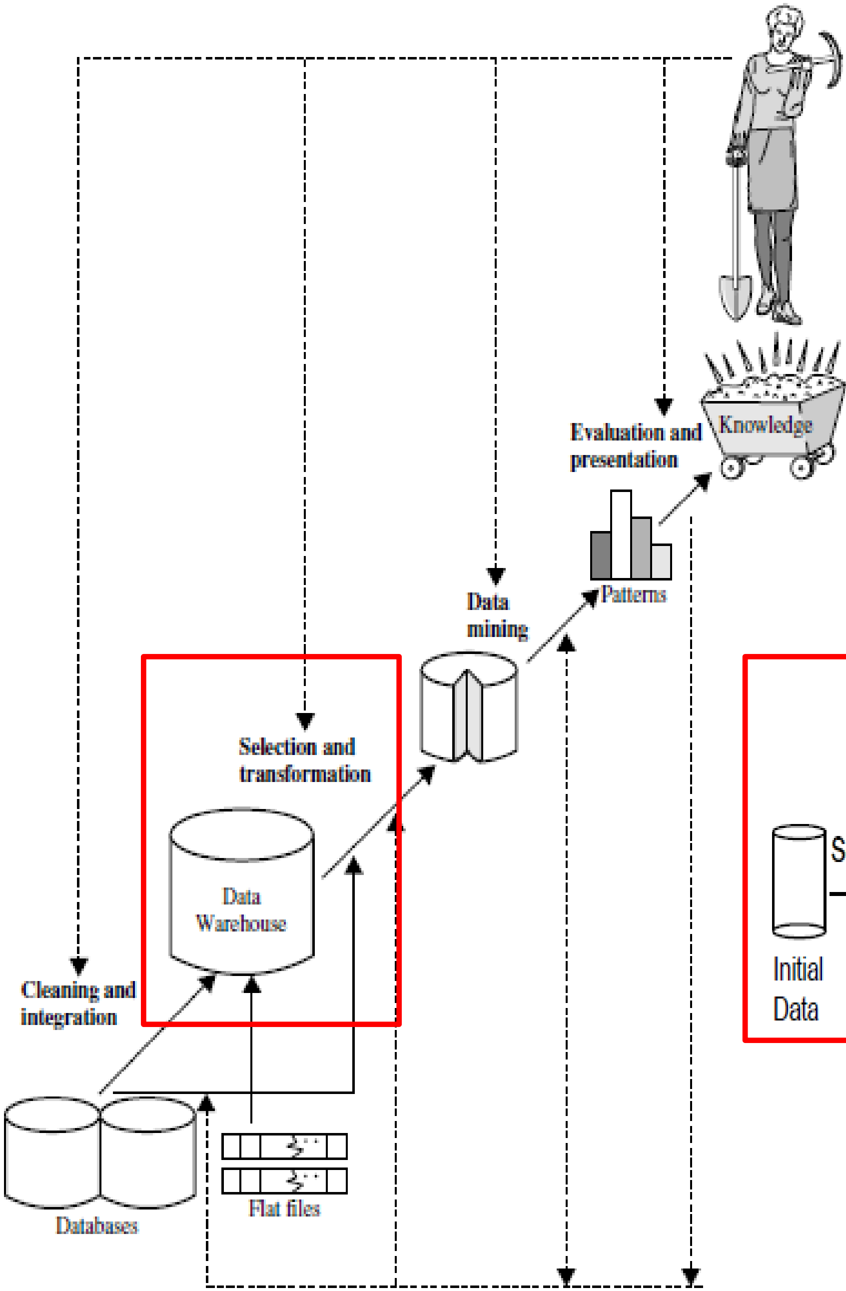
- Process of semi-automatically analyzing large databases to find patterns that are:
  - valid: hold on new data with some certainty
  - novel: non-obvious to the system
  - useful: should be possible to act on the item
  - understandable: humans should be able to interpret the pattern

- MHD
  1. Human Interaction
  2. Overfitting
  3. Outliers

DM Issues ?

1. Interpretation of results

1. Visualization of results
2. Large Datasets
3. High Dimensionality



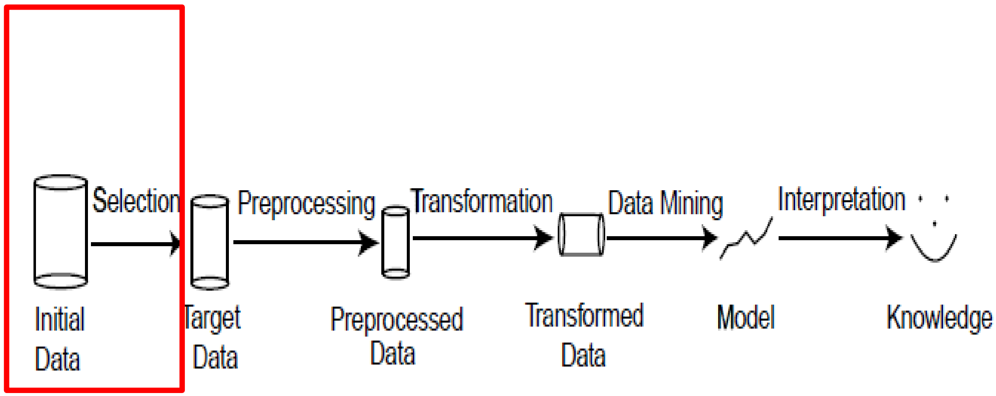
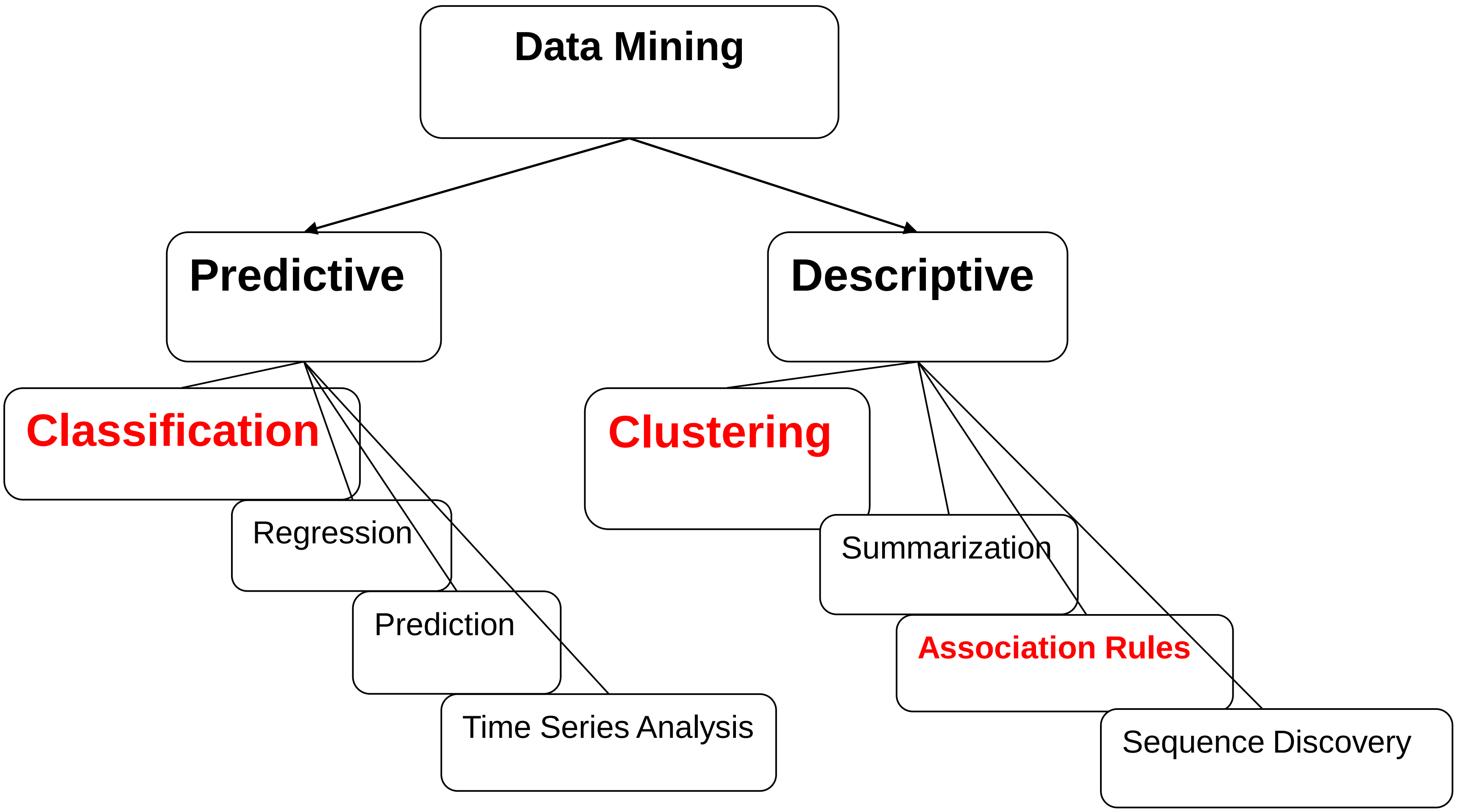


Figure 1.3: KDD Process

Figure 1.4 Data mining as a step in the process of knowledge discovery.

Data Mining Tasks

(Styles of learning)



Basic Concepts

- **Classification is a form of data analysis that extracts models describing important data classes.**

- Data classification is a two-step process, consisting of
    - A **learning step ,** where a classification model is constructed and
    - A **classification step,** where the model is used to predict class labels for given data

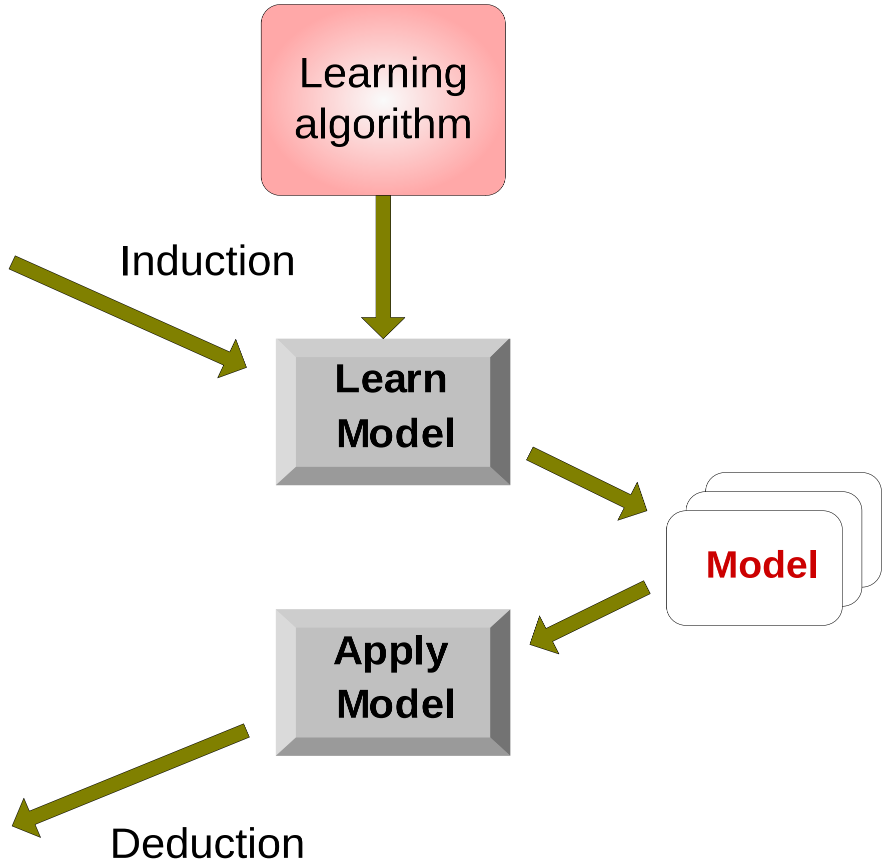
Basic Concepts

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Basic Concepts

- In the first step, a classifier is built : this is the **learning step (or training phase),** where a classification algorithm builds the classifier by analyzing or "learning from" a training set made up of database tuples and their associated class labels.
- A tuple, *X, is represented by an n-dimensional* attribute vector, *X ={x1, x2, …, xn} depicting n measurements made on the tuple* from *n database attributes, respectively, A1, A2, :…, An.*
- *Each tuple, X, is assumed to* belong to a predefined class as determined by another database attribute called the **class label attribute.**
- The class label attribute is discrete-valued and unordered. It is *categorical* (or nominal) in that each value serves as a category or class.
- *The individual tuples making up the training set are referred to as training tuples and are randomly sampled from the database under analysis. In the context of classification, data tuples can be referred to as samples, examples, instances, data points, or objects.*

Basic Concepts

- The class label of each training tuple is provided, this step is also known as **supervised learning** (i.e., the learning of the classifier is "supervised" in that it is told to which class each training tuple belongs).

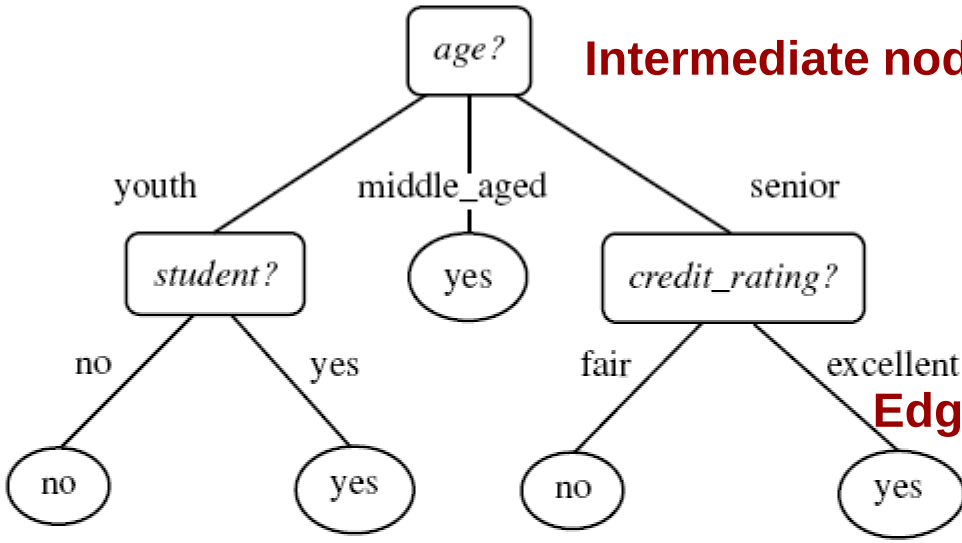
- It contrasts with **unsupervised learning (or clustering)**, in which the class label of each training tuple is not known, and the number or set of classes to be learned may not be known in advance.

- This first step of the classification process can also be viewed as the **learning of a mapping or function**, $y = f(X)$, that can predict the associated class label y of a given tuple X.

- The **accuracy** of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier.

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 (nonleaf 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*.**
- *The topmost node in* a tree is the root node.
- Internal nodes are denoted by rectangles, and
- Leaf nodes are denoted by ovals.



Intermediate nodes : Attributes

Edges : Attribute value tests

Leaf nodes : Class predictions

Figure : A decision tree for the concept *buys computer, e*ach leaf node represents a class :
buys computer = yes or buys computer = no

Example algorithms: ID3, C4.5,CART

Decision Tree Induction

- *"How are decision trees used for classification?"*
- *Given a tuple, **X, for which the associated*** class label is unknown, the attribute values of the tuple are tested against the decision tree.
- A path is traced from the root to a leaf node, which holds the class prediction for that tuple.
- **Decision trees can easily be converted to classification rules.**

Decision Tree Induction

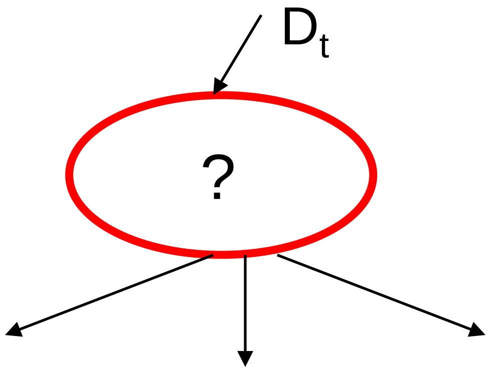
- During the late 1970s and early 1980s, J. Ross Quinlan, a researcher in machine learning, developed a decision tree algorithm known as **ID3 (Iterative Dichotomiser).**
- In 1984, a group of statisticians (L. Breiman, J. Friedman, R. Olshen, and C. Stone) published the book **Classification and Regression Trees (CART),** which described the generation of binary decision trees.
- ID3 and CART were invented independently of one another at around the same time, yet follow a similar approach for learning decision trees from training tuples.
- ID3, C4.5, and CART adopt a **greedy** (i.e., non-backtracking) approach in which decision trees are constructed in a top-down recursive divide-and-conquer manner.
- The training set is recursively partitioned into smaller subsets as the tree is being built.

Decision Tree Induction

- The algorithm is called with three parameters:
  - **D, attribute list, and Attribute selection method.**
  -
- *D as a data partition. Initially, it is the complete set* of training tuples and their associated class labels.
- The parameter *attribute list is a* 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.
- This procedure employs an attribute selection measure such as **Information Gain or the Gini index**.

DT: Algorithm

- Let D_t be the set of training records that reach a node t
- General Procedure:
  - If D_t contains records that belong the same class y_t, then t is a leaf node labeled as y_t
  - If D_t is an empty set, then t is a leaf node labeled by the default class, y_d
  - If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets. Recursively apply the procedure to each subset.



Issues

- Determine how to split the records
  - How to specify the attribute test condition?
  - How to determine the best split?
- Determine when to stop splitting

DT: Algorithm

- **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.**
- **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) .....

DT: Algorithm

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.

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 ← 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;
```

DT: Algorithm

- The recursive partitioning stops only when any one of the following terminating conditions is true:

1. **All the tuples in partition D (represented at node N) belong to the same class (steps 2 and 3)**.
2. **There are no remaining attributes on which the tuples may be further partitioned** (step 4).
    1. In this case, majority voting is employed (step 5). This involves converting node N into a leaf and labeling it with the most common class in D. Alternatively, the class distribution of the node tuples may be stored.
3. **There are no tuples for a given branch, that is, a partition *Dj is empty (step 12).***
    1. In this case, a leaf is created with the majority class in *D (step 13).*

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.
- The attribute selection measure provides a ranking for each attribute describing the given training tuples. The attribute having the best score for the measure is chosen as the *splitting attribute for the given tuples.*
- Three popular attribute selection measures
  - information gain, gain ratio, and Gini index

Information Gain

- ID3 uses **information gain as its attribute selection measure.**
- **This measure is based on** pioneering work by Claude Shannon on information theory.
- Let node N represent or hold the tuples of partition D.
- The attribute with the highest information gain is chosen as the splitting attribute for node N.
- This attribute minimizes the information needed to classify the tuples in the resulting partitions and reflects the least randomness or “impurity” in these partitions.
- Such an approach minimizes the expected number of tests needed to classify a given tuple and guarantees that a simple tree is found.

Attribute Selection Measures

- The notation
- Let D, the data partition, be a training set of class-labeled tuples.
- Suppose the class label attribute has m distinct values defining m distinct classes, C_i (for i= 1, …, m).
- Let $C_{i,D}$ be the set of tuples of class Ci in D. Let $|D|$ and $|C_{i,D}|$ denote the number of tuples in D and $C_{i,D}$, respectively.

Information Gain

- The expected information needed to classify a tuple in *D is given by*

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

- where pi is the nonzero probability that an arbitrary tuple in D belongs to class Ci and is estimated by $|C_{i,D}|/|D|$.
- A log function to the base 2 is used, because the information is encoded in bits.
- Info(D) is just the average amount of information needed to identify the class label of a tuple in D.
- Note that, at this point, the information we have is based solely on the proportions of tuples of each class.

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

Attribute Selection Measures

- How much more information would we still need (after the partitioning) to arrive at an exact classification? This amount is measured by

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

- The term $|D_j|/\ |D|$ acts as the weight of the jth partition.
- $Info_A(D)$ is the expected information required to classify a tuple from D based on the partitioning by A.
- The smaller the expected information (still) required, the greater the purity of the partitions.
- Information gain is defined as the difference between the original information
- Requirement and the new requirement (i.e., obtained after partitioning on A).
- That is, $Gain(A) = Info(D) – Info_A(D)$
  - In other words, Gain(A) tells us how much would be gained by branching on A.

Attribute Selection Measures

- Select the attribute with the highest information gain
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:

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

- Information needed (after using A to split D into v partitions) to classify D:

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

- Information gained by branching on attribute A

$$\text{Gain}(A) = \text{Info}(D) - \text{Info}_A(D)$$

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

Example

- The class label attribute, buys computer, has two distinct values, namely, {yes, no}; therefore, there are two distinct classes (i.e., m = 2).
- Let class C1 correspond to yes and class C2 correspond to no.
- There are nine tuples of class yes and five tuples of class no.
- A (root) node N is created for the tuples in D.
- To find the splitting criterion for these tuples, we must compute the information gain of each attribute.
- We first compute the expected information needed to classify a tuple in D

Buys Computer	
Yes	No
9	5

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

Example

- Next, we need to compute the expected information requirement for each attribute.
- Let's start with the attribute age.
- We need to look at the distribution of yes and no tuples for each category of age.
- For the age
  - category "youth," there are two yes tuples and three no tuples.
  - category "middle aged," there are four yes tuples and zero no tuples.
  - category "senior," there are three yes tuples and two no tuples.
- Compute the expected information needed to classify a tuple in D if the tuples are partitioned according to age is

		Buys Computer		
		Yes	No	
Age	Youth	2	3	5
	middle_aged	4	0	4
	Senior	3	2	5
				14

$$Info_{age}(D) = \frac{5}{14} \times \left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) + \frac{4}{14} \times \left(-\frac{4}{4}\log_2\frac{4}{4}\right) + \frac{5}{14} \times \left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right)$$

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

AVC-set on *Student*

student	Buy_Computer	
	yes	no
yes	6	1
no	3	4

AVC-set on *credit_rating*

Credit rating	Buy_Computer	
	yes	no
fair	6	2
excellent	3	3

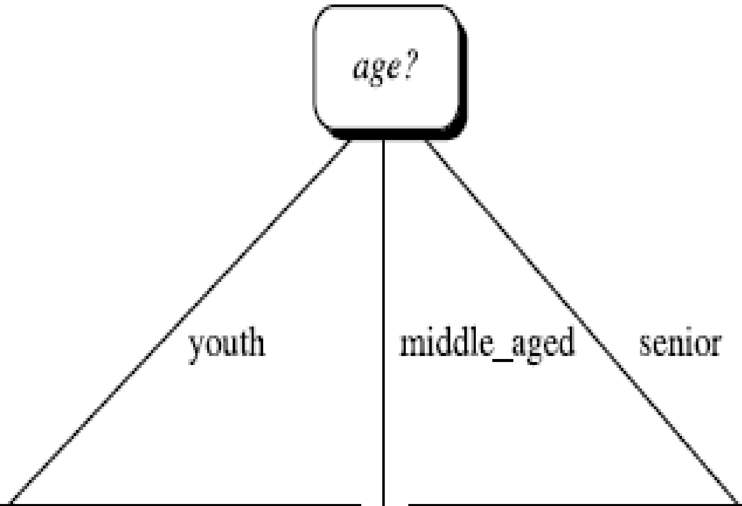
AVC-set on *income*

income	Buy_Computer	
	yes	no
high	2	2
medium	4	2
low	3	1

Example

- Hence, the gain in information from such a partitioning would be
- *Gain(age) = Info(D)- Info$_{age}$(D)*
- *= 0.940- 0.694*
- *= 0.246 bits.*

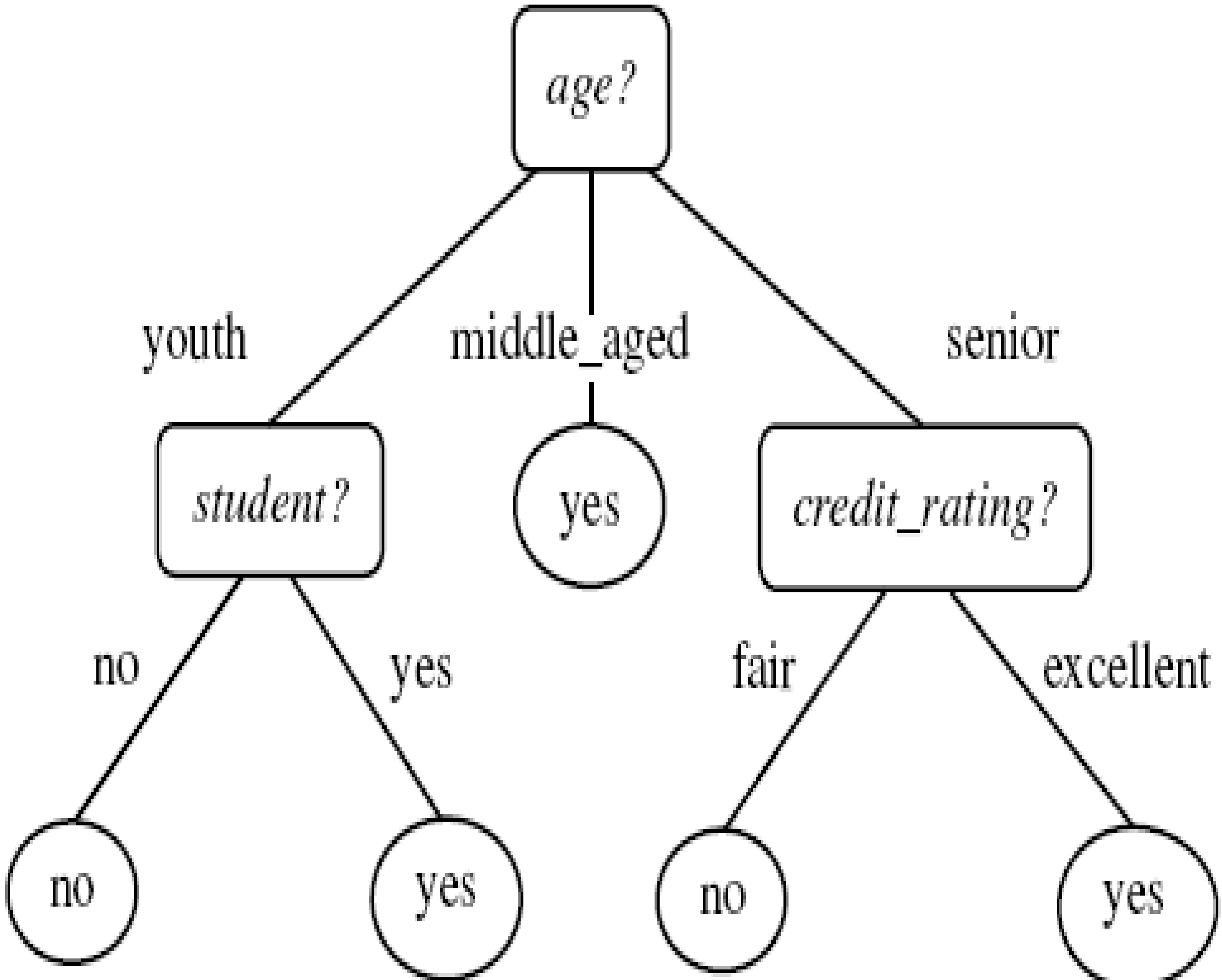
Similarly, we can compute $\boxed{Gain(income) = 0.029 \text{ bits},}$ $\boxed{Gain(student) = 0.151 \text{ bits},}$ and $\boxed{Gain(credit\_rating) = 0.048}$ bits. Because *age* has the highest information gain among the attributes, it is selected as the splitting attribute. Node *N* is labeled with *age*,



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*
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

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


age?
youth
middle_aged
senior
student?
yes
credit_rating?
no
yes
fair
excellent
no
yes
no
yes

Viewer

Relation: weather

No.	outlook Nominal	temperature Numeric	humidity Numeric	windy Nominal	play Nominal
1	sunny	85.0	85.0	FALSE	no
10	rainy	75.0	80.0	FALSE	yes
11	sunny	75.0	70.0	TRUE	yes
12	overcast	72.0	90.0	TRUE	yes
13	overcast	81.0	75.0	FALSE	yes
14	rainy	71.0	91.0	TRUE	no
2	sunny	80.0	90.0	TRUE	no
3	overcast	83.0	86.0	FALSE	yes
4	rainy	70.0	96.0	FALSE	yes
5	rainy	68.0	80.0	FALSE	yes
6	rainy	65.0	70.0	TRUE	no
7	overcast	64.0	65.0	TRUE	yes
8	sunny	72.0	95.0	FALSE	no
9	sunny	69.0	70.0	FALSE	yes

Viewer

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	play Nominal
1	sunny	hot	high	FALSE	no
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes

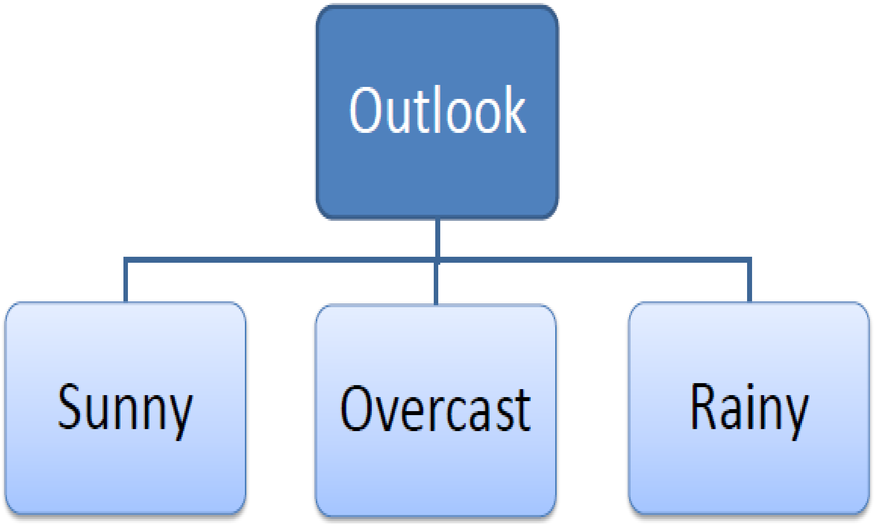
Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

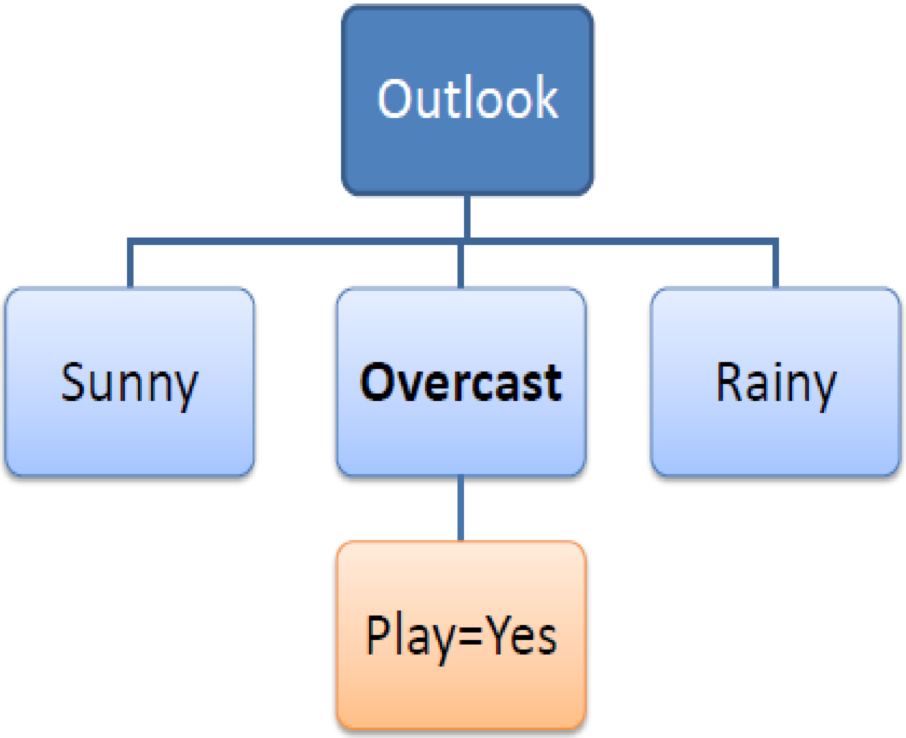
		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

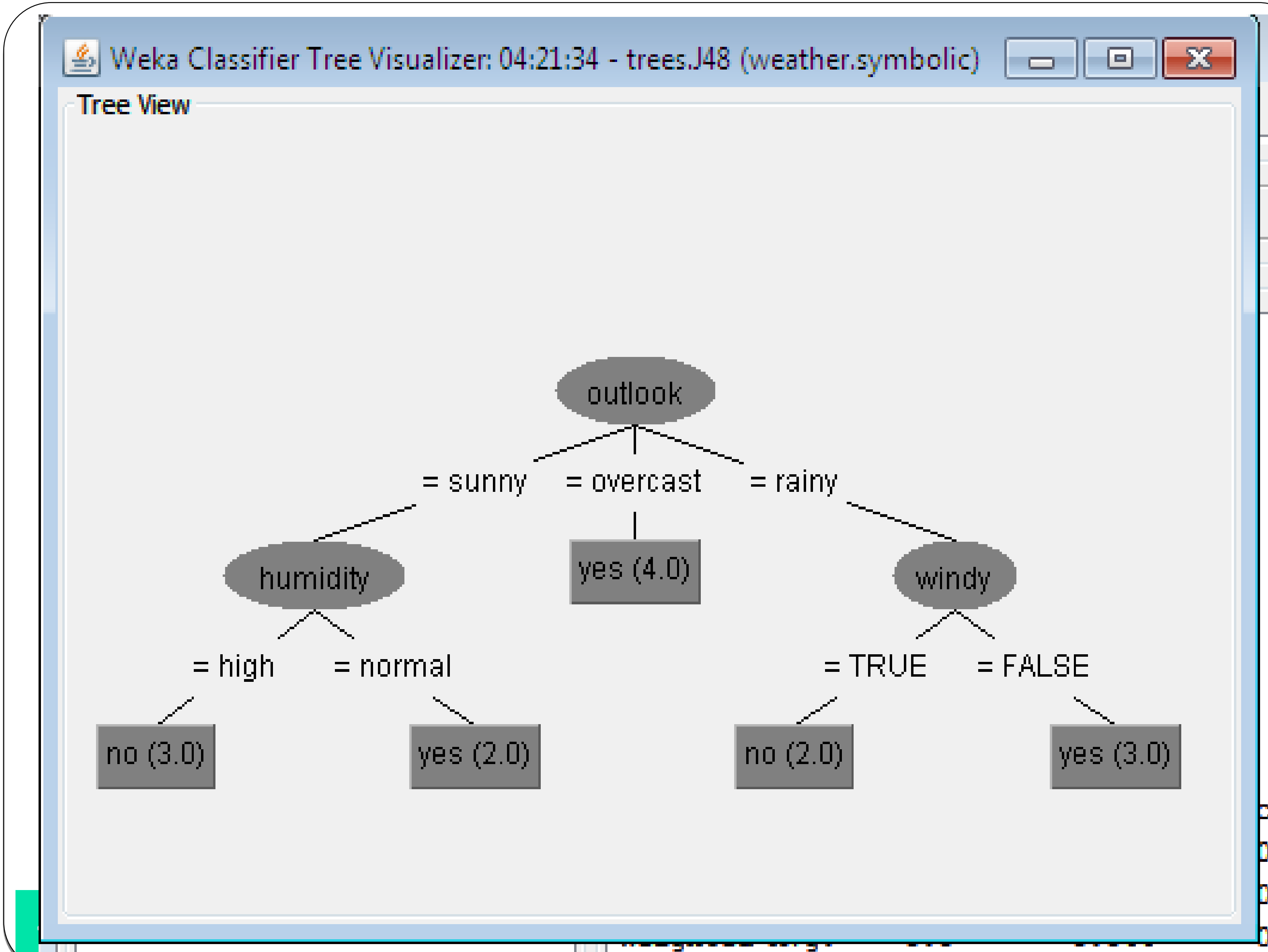
		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

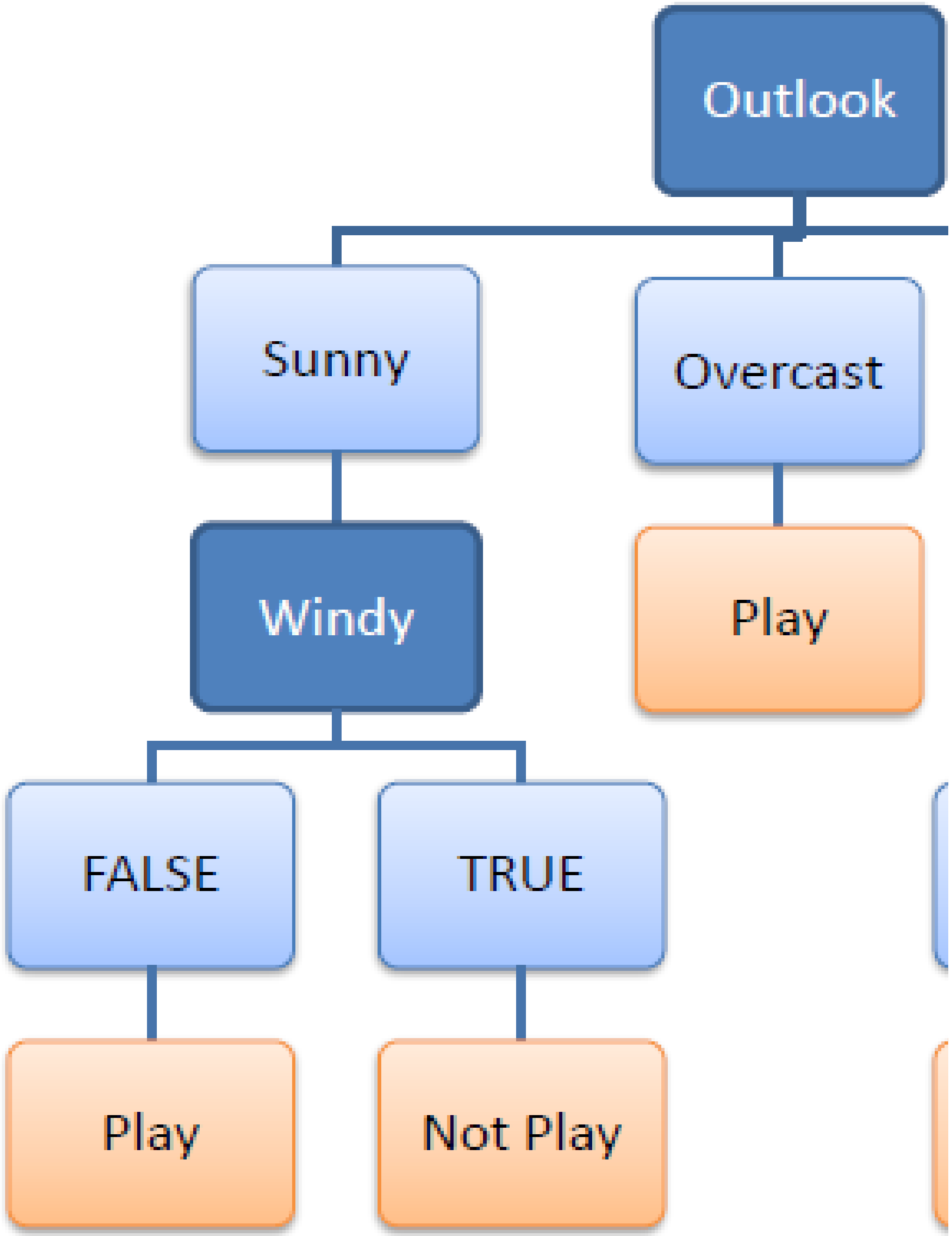
		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			


Outlook
Sunny
Overcast
Rainy


Outlook
Sunny
Overcast
Rainy
Play=Yes



Weka Classifier Tree Visualizer: 04:21:34 - trees.J48 (weather.symbolic)
Tree View
outlook
= sunny
= overcast
= rainy
humidity
yes (4.0)
windy
= high
= normal
= TRUE
= FALSE
no (3.0)
yes (2.0)
no (2.0)
yes (3.0)



R_1: IF (Outlook=Sunny) AND (Windy=FALSE) THEN Play=Yes

R_2: IF (Outlook=Sunny) AND (Windy=TRUE) THEN Play=No

R_3: IF (Outlook=Overcast) THEN Play=Yes

R_4: IF (Outlook=Rainy) AND (Humidity=High) THEN Play=No

R_5: IF (Outlook=Rain) AND (Humidity=Normal) THEN Play=Yes

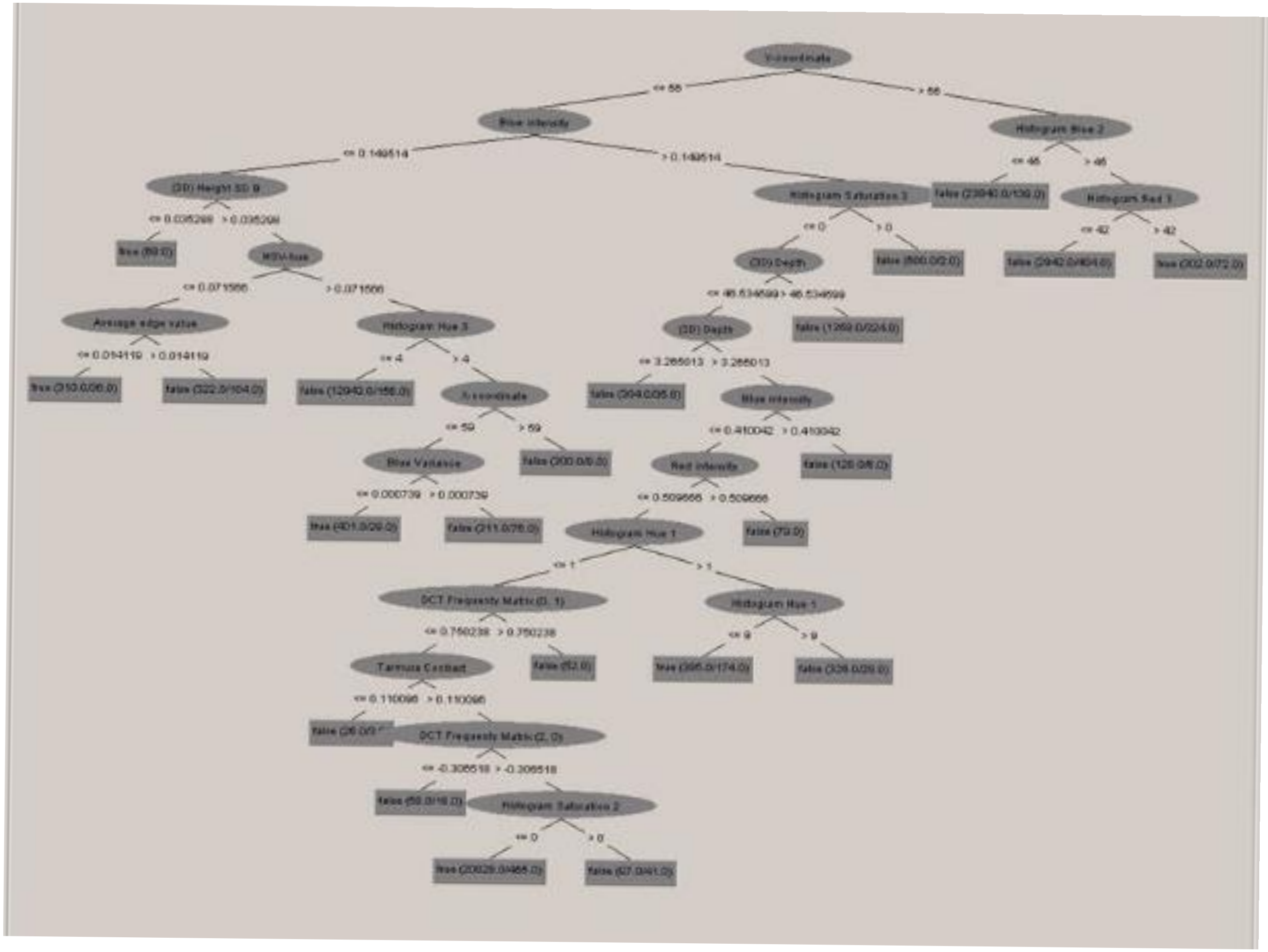
5 (a) A simple example from the stock market involving only discrete ranges has Profit as categorically attribute, with values {up, down} and the training data is 10 M

AGE	COMPETITION	TYPE	PROFIT
Old	Yes	Software	Down
Old	No	Software	Down
Old	No	Hardware	Down
Mid	Yes	Software	Down
Mid	Yes	Hardware	Down
Mid	No	Hardware	Up
Mid	No	Software	Up
New	Yes	Software	Up
New	No	Hardware	Up
New	No	Software	Up

Apply decision tree algorithm and show the generated rules.

Classification

DT



Pruning Trees

1. The main drawback of DT is overfitting.
   The generated tree may overfit the training data

   1. Too many branches,some may reflect anamolies due to noise and outlier
   2.poor accuracy for unseen samples
   Two approches to avoid overfitting

   1. Prepruning: The incorrect branches are discarded by halting tree construction.

      1.Do not split a node if this would result in the goodness measure falling below a threshold value, otherwise it is grown
      2.Prepruning:Early stopping

   2.Postpruning:Grow the whole tree then prune subtrees which overfit on the pruning set

   Prepruning is faster, postpruning is more accurate (requires a separate pruning set)

   Most popular test: chi-squared test

   ID3 used chi-squared test in addition to information gain
   Only statistically significant attributes were allowed to be selected by information gain procedure

Naïve Bayes

Basic Concepts

- Bayesian classifiers are statistical classifiers.
- They can predict class membership probabilities such as the probability that a given tuple belongs to a particular class.

- Bayesian classification is based on Bayes' theorem.
- Naive Bayesian classifiers assume that the effect of an attribute value on a given class is independent of the values of the other attributes.
- This assumption is called **class conditional independence**. It is made to simplify the computations involved and, in this sense, is considered "naive."

Bayes' Theorem

- Let X be a data tuple.
- In Bayesian terms, X is considered "evidence."
- As usual, it is described by measurements made on a set of n attributes.
- Let H be some hypothesis such as that the data tuple X belongs to a specified class C.
- For classification problems, we want to determine P(H|X), the probability that the hypothesis H holds given the "evidence" or observed data tuple X.
- In other words, we are looking for the probability that tuple X belongs to class C, given that we know the attribute description of X.

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}.$$

Bayes' Theorem

- P(H|X) is the posterior probability, or a posteriori probability, of H conditioned on X.
- For example, suppose our world of data tuples is confined to customers described by the attributes age and income, respectively, and that X is a 35-year-old customer with an income of $40,000.
- Suppose that H is the hypothesis that our customer will buy a computer.
- Then P(H|X) reflects the probability that customer X will buy a computer given that we know the customer's age and income.

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}.$$

Naive Bayesian Classification

- The naïve Bayesian classifier, or simple Bayesian classifier, works as follows:

1. Let D be a training set of tuples and their associated class labels. As usual, each tuple is represented by an n-dimensional attribute vector, X ={ x1, x2, : : : , xn}, depicting n measurements made on the tuple from n attributes, respectively, A1, A2, : : : , An.
2. Suppose that there are m classes, C1, C2, : : : , Cm. Given a tuple, X, the classifier will predict that X belongs to the class having the highest posterior probability, conditioned on X. That is, the na¨ıve Bayesian classifier predicts that tuple X belongs to the class Ci if and only if

$$P(C_i|X) > P(C_j|X) \quad \text{for } 1 \leq j \leq m, j \neq i.$$

Naïve Bayesian Classification

2. Thus, we maximize P(Ci|X). The class Ci for which P(Ci|X) is maximized is called the maximum posteriori hypothesis. By Bayes' theorem

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}.$$

3. As P(X) is constant for all classes, only P(X|Ci)/P(Ci) needs to be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, that is, P(C1)= P(C2) =, …, = P(Cm), and we would therefore maximize P(X|Ci).

Naïve Bayesian Classification

4. Given data sets with many attributes, it would be extremely computationally expensive to compute P(X|Ci). To reduce computation in evaluating P(X|Ci), the na¨ıve assumption of class-conditional independence is made. This presumes that the attributes' values are conditionally independent of one another, given the class label of the tuple (i.e., that there are no dependence relationships among the attributes). Thus,

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i)$$
$$= P(x_1|C_i) \times P(x_2|C_i) \times \cdots \times P(x_n|C_i).$$

Naïve Bayesian Classification

5. To predict the class label of X, P(X|Ci)P(Ci) is evaluated for each class Ci . The classifier predicts that the class label of tuple X is the class Ci if and only if

$$P(X|C_i)P(C_i) > P(X|C_j)P(C_j) \quad \text{for } 1 \le j \le m, j \ne i.$$

In other words, the predicted class label is the class Ci for which P(**X|Ci)/P(Ci) is the** maximum.

Naïve Bayesian Classification

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

Example

- The data tuples are described by the attributes age, income, student, and credit rating.
- The class label attribute, buys computer, has two distinct values namely, {yes, no}).
- Let C1 correspond to the class buys computer D yes and C2 correspond to buys computer D no. The tuple we wish to classify is

$$X = (age = youth, income = medium, student = yes, credit\_rating = fair)$$

Example

- We need to maximize $P(X|Ci)P(Ci)$, for $i=1, 2$.
- $P(Ci)$, the prior probability of each class, can be computed based on the training tuples:

$$P(buys\_computer = yes) = 9/14 = 0.643$$
$$P(buys\_computer = no) = 5/14 = 0.357$$

To compute $P(X|C_i)$, for $i = 1, 2$, we compute the following conditional probabilities:

$$P(age = youth \mid buys\_computer = yes) = 2/9 = 0.222$$
$$P(age = youth \mid buys\_computer = no) = 3/5 = 0.600$$
$$P(income = medium \mid buys\_computer = yes) = 4/9 = 0.444$$
$$P(income = medium \mid buys\_computer = no) = 2/5 = 0.400$$
$$P(student = yes \mid buys\_computer = yes) = 6/9 = 0.667$$
$$P(student = yes \mid buys\_computer = no) = 1/5 = 0.200$$

Example

$$P(credit\_rating = fair \mid buys\_computer = yes) = 6/9 = 0.667$$
$$P(credit\_rating = fair \mid buys\_computer = no) = 2/5 = 0.400$$

Using these probabilities, we obtain

$$\begin{aligned} P(X \mid buys\_computer = yes) = {} & P(age = youth \mid buys\_computer = yes) \\ & \times P(income = medium \mid buys\_computer = yes) \\ & \times P(student = yes \mid buys\_computer = yes) \\ & \times P(credit\_rating = fair \mid buys\_computer = yes) \\ = {} & 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044. \end{aligned}$$

Similarly,

$$P(X \mid buys\_computer = no) = 0.600 \times 0.400 \times 0.200 \times 0.400 = 0.019.$$

To find the class, *Ci , that maximizes P(**X|Ci)P(Ci), we compute***

$$P(X \mid buys\_computer = yes)P(buys\_computer = yes) = 0.044 \times 0.643 = 0.028$$
$$P(X \mid buys\_computer = no)P(buys\_computer = no) = 0.019 \times 0.357 = 0.007$$

Therefore, the naïve Bayesian classifier predicts $buys\_computer = yes$ for tuple X.

Example

CHILLS	RUNNY NOSE	HEADACHE	FEVER	FLU?
Y	N	Mild	Y	N
Y	Y	No	N	Y
Y	N	Strong	Y	Y
N	Y	Mild	Y	Y
N	N	No	N	N
N	Y	Strong	Y	Y
N	Y	Strong	N	N
Y	Y	Mild	Y	Y

Data sample X = (chills=Y, runnynose=N, headache=mild, fever=Y)

For given data sample find whether the person has flu or no using Naive Bayesian classification.

Example

C1 :Flu = “yes” C2 : Flu = “no” The total number of records in the table are 8.

P(Ci) = P(Flu = “yes”) = 5/8 = 0.62

P(Flu = “no”) = 3/10= 0.37

Compute P(X|Ci) for each class

P(chills=“Y” | Flu = “yes”) = 3/5 = 0.6

P(chills =“Y” |Flu = “no”) = 1/3=0.33

P(runny nose=“N” | Flu = “yes”) = 1/5 = 0.2

P(runny nose=“N” |Flu = “no”) =2/3 = 0.66

P(headache=“mild” | Flu = “yes”) = 2/5 = 0.4

P(headache=“mild” |Flu = “no”) = 1/3 = 0.33

P(fever=“Y” | Flu = “yes”)= 4/5= 0.8

P(fever=“Y” |Flu = “no”) = 1/3=0.33

Example

Multiplying all the probabilities with yes values and no values from above separately.

P(X|Ci) :
P(X|Flu= "yes")
= 0.6 x 0.4 x 0.2 x 0.8
= 0.0384
P(X|Flu = "no")
= 0.33x 0.66 x 0.33 x 0.33
= 0.024
Now, multiply these probabilities with the above calculated P(Ci)

P(X|Ci)*P(Ci) :
P(X|Flu = "yes") * P(Flu = "yes") = 0.0384 x 0.62=0.023
P(X|Flu = "no") * P(Flu = "no") = 0.024 x0.37=0.0088

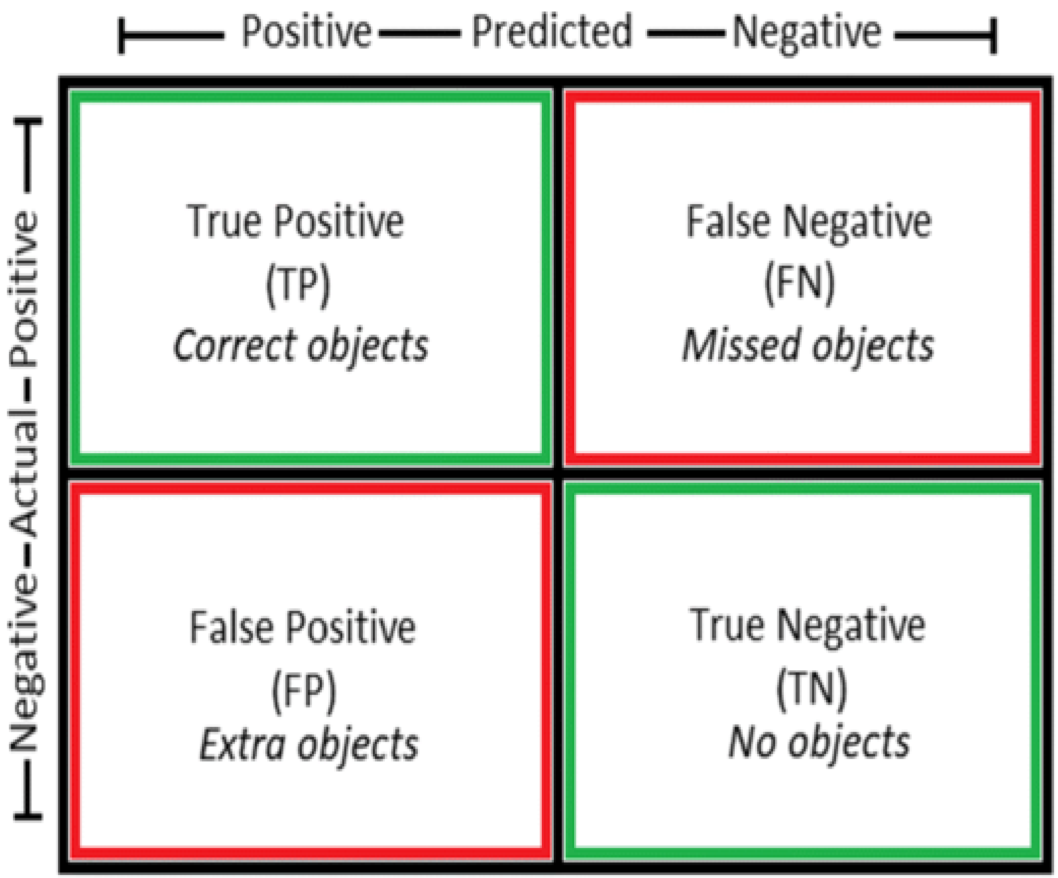
The maximum value is 0.023.
Therefore, X belongs to class ("Flu=yes")

Example

PATIENT	DISEASE	SUGAR LEVEL	SURVIVAL CHANCES
Small	Serious	High	Yes
Medium	Normal	Low	Yes
Senior	Lifetime	Normal	Yes
Small	Lifetime	High	No
Small	Normal	High	Yes
Senior	Serious	Normal	No
Medium	Serious	Low	Yes
Senior	Normal	Low	No
Medium	Lifetime	Normal	Yes
Medium	Serious	High	No
Senior	Normal	Low	No

Data sample X = (age=senior, disease=normal, sugar level=normal)

Confusion matrix and performance evaluation measures



Recall

$$\frac{TP}{(TP+FN)}$$

$$Accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)}$$

Specificity

$$\frac{TN}{(TN+FP)}$$

$$F1 = \frac{2 \text{ x Recall x Precision}}{\text{Recall + Precision}}$$

Precision

$$\frac{TP}{(TP+FP)}$$

Negative predictive value

$$\frac{TN}{(TN+TP)}$$

Confusion matrix and performance evaluation measures

Recall: The ability of a model to find all the relevant cases within a data set. Mathematically, we define recall as the number of true positives divided by the number of true positives plus the number of false negatives.

Precision: The ability of a classification model to identify only the relevant data points. Mathematically, precision the number of true positives divided by the number of true positives plus the number of false positives.

classification model predicting emails as "spam" or "normal"

	Predicted class POSITIVE (spam)	Predicted class NEGATIVE (normal)
Actual class POSITIVE (spam)	TRUE POSITIVE (TP) 320	FALSE NEGATIVE (FN) 43
Actual class NEGATIVE (normal)	FALSE POSITIVE (FP) 20	TRUE NEGATIVE (TN) 538

Distance function

Distance functions

Euclidean $$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan $$\sum_{i=1}^{k}|x_i - y_i|$$

Minkowski $$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$

For p=2, we get the L2 form, which is Manhattan Distance
p=1, we get the L1 form, which is Euclidean Distance

Euclidean Distance

Point	X	Y
P1	0	2
P2	2	0
P3	3	1
P4	5	1

Calculate the Distance Matrix

Distance function

Given two objects represented by the tuples (15, 7, 24, 21) and (12, 0, 16, 10);

Compute the Euclidean distance between the two objects.
(b) Compute the Manhattan distance between the two objects
(c) Compute the Minkowski distance between the two objects, using h=3.
(d) Compute the supremum distance between the two objects

(c) Minkowski Distance:

$$d(i,j) = \sqrt[3]{|15-12|^3 + |7-0|^3 + |24-16|^3 + |21-10|^3}$$

$$= \sqrt[3]{(3)^3 + (7)^3 + (8)^3 + (11)^3}$$

$$= \sqrt[3]{27 + 343 + 512 + 1331}$$

$$= \sqrt[3]{2213} = 13.03 \quad \underline{\text{Ans}}$$

(d) Supremum distance:

$$d(i,j) = \max(|x_{if} - x_{jf}|)$$

$$= |24-16|$$

$$= 8 \quad \underline{\text{Ans}}$$