2nd approach: Gain Ratio

- **Problem of Information gain approach**
  - Biased towards tests with many outcomes (attributes having a large number of values)
  - E.g: attribute acting as unique identifier
    - Produce a large number of partitions (1 tuple per partition)
    - Each resulting partition D is pure \( \text{Info}(D)=0 \)
    - The information gain is maximized

- **Extension to Information Gain**
  - C4.5, a successor of ID3 uses an extension to information gain known as **gain ratio**
  - Overcomes the bias of Information gain
  - Applies a kind of normalization to information gain using a split information value
2nd approach: Gain Ratio

- The **split information value** represents the potential information generated by splitting the training data set \( D \) into \( v \) partitions, corresponding to \( v \) outcomes on attribute \( A \)

\[
SplitInfo_A(D) = - \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2 \left( \frac{|D_j|}{|D|} \right)
\]

- **High splitInfo**: partitions have more or less the same size (uniform)
- **Low split Info**: few partitions hold most of the tuples (peaks)

- The gain ratio is defined as

\[
GainRatio (A) = \frac{Gain(A)}{SplitInfo(A)}
\]

- The attribute with the maximum gain ratio is selected as the splitting attribute
## Gain Ratio: Example

<table>
<thead>
<tr>
<th>RID</th>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit-rating</th>
<th>class:buy_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>youth</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>youth</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td>middle-aged</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>senior</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>5</td>
<td>senior</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>6</td>
<td>senior</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>7</td>
<td>middle-aged</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>8</td>
<td>youth</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>9</td>
<td>youth</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>10</td>
<td>senior</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>11</td>
<td>youth</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>12</td>
<td>middle-aged</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>13</td>
<td>middle-aged</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>14</td>
<td>senior</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
</tbody>
</table>

Using attribute income

1\textsuperscript{st} partition (low) \( D_1 \) has 4 tuples
2\textsuperscript{nd} partition (medium) \( D_2 \) has 6 tuples
3\textsuperscript{rd} partition (high) \( D_3 \) has 4 tuples

Gain (income) = 0.029

GainRatio (income) = \( \frac{0.029}{0.926} \) = 0.031

\[
\text{SplitInfo}_{\text{income}}(D) = - \frac{4}{14} \log_2 \left( \frac{4}{14} \right) - \frac{6}{14} \log_2 \left( \frac{6}{14} \right) - \frac{4}{14} \log_2 \left( \frac{4}{14} \right) \\
= 0.926
\]
The Gini Index (used in CART) measures the impurity of a data partition $D$:

$$Gini\ (D) = 1 - \sum_{i=1}^{m} p_i^2$$

- $m$: the number of classes
- $p_i$: the probability that a tuple in $D$ belongs to class $C_i$

The Gini Index considers a **binary split** for each attribute $A$, say $D_1$ and $D_2$. The Gini index of $D$ given that partitioning is:

$$Gini_A\ (D) = \frac{D_1}{D} Gini\ (D_1) + \frac{D_2}{D} Gini\ (D_2)$$

- A weighted sum of the impurity of each partition

The reduction in impurity is given by

$$\Delta Gini\ (A) = Gini\ (D) - Gini_A\ (D)$$

The attribute that maximizes the reduction in impurity is chosen as the splitting attribute
Binary Split: Continuous-Valued Attributes

- **D**: a data partition
- Consider attribute **A** with continuous values
- To determine the best binary split on **A**

**What to examine?**
- Examine each possible split point
- The midpoint between each pair of (sorted) adjacent values is taken as a possible split-point

**How to examine?**
- For each split-point, compute the weighted sum of the impurity of each of the two resulting partitions (D1: A <= split-point, D2: A > split-point)

\[
Gini_A(D) = \frac{D_1}{D} Gini(D_1) + \frac{D_2}{D} Gini(D_2)
\]

- The split-point that gives the minimum Gini index for attribute **A** is selected as its splitting subset
Binary Split: Discrete-Valued Attributes

- **D**: a data partition
- Consider attribute **A** with **v** outcomes \( \{a_1...a_v\} \)
- To determine the best binary split on **A**

**What to examine?**
- Examine the partitions resulting from all possible subsets of \( \{a_1...a_v\} \)
- Each subset \( S_A \) is a binary test of attribute **A** of the form “**A** \( \in S_A \) ?”
- \( 2^v \) possible subsets. We exclude the power set and the empty set, then we have \( 2^v - 2 \) subsets

**How to examine?**
- For each subset, compute the weighted sum of the impurity of each of the two resulting partitions

\[
Gini_A(D) = \frac{D_1}{D} Gini(D_1) + \frac{D_2}{D} Gini(D_2)
\]

- The subset that gives the minimum Gini index for attribute **A** is selected as its splitting subset
Gini(income)

<table>
<thead>
<tr>
<th>RID</th>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit-rating</th>
<th>class: buy_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>youth</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>youth</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td>middle-aged</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>senior</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>5</td>
<td>senior</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>6</td>
<td>senior</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>7</td>
<td>middle-aged</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>8</td>
<td>youth</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>9</td>
<td>youth</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>10</td>
<td>senior</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>11</td>
<td>youth</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>12</td>
<td>middle-aged</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>13</td>
<td>middle-aged</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>14</td>
<td>senior</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
</tbody>
</table>

Compute the Gini index of the training set $D$: 9 tuples in class yes and 5 in class no

$$Gini(D) = 1 - \left( \left( \frac{9}{14} \right)^2 + \left( \frac{5}{14} \right)^2 \right) = 0.459$$

Using attribute income: there are three values: low, medium and high

Choosing the subset $\{\text{low, medium}\}$ results in two partitions:

- $D1 (income \in \{\text{low, medium}\})$: 10 tuples
- $D2 (income \in \{\text{high}\})$: 4 tuples
The Gini Index measures of the remaining partitions are:

\[
Gini_{\{low, medium\} \rightarrow \{high\}}(D) = 0.450
\]

Therefore, the best binary split for attribute income is on \{medium, high\} and \{low\}.
Comparing Attribute Selection Measures

The three measures, in general, return good results but

- **Information Gain**
  - Biased towards multivalued attributes

- **Gain Ratio**
  - Tends to prefer unbalanced splits in which one partition is much smaller than the other

- **Gini Index**
  - Biased towards multivalued attributes
  - Has difficulties when the number of classes is large
  - Tends to favor tests that result in equal-sized partitions and purity in both partitions
2.2.3 Tree Pruning

- **Problem: Overfitting**
  - Many branches of the decision tree will reflect anomalies in the training data due to noise or outliers
  - Poor accuracy for unseen samples

- **Solution: Pruning**
  - Remove the least reliable branches
Tree Pruning Approaches

- **1st approach: prepruning**
  - Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
  
  - Statistical significance, information gain, Gini index are used to assess the goodness of a split
  
  - Upon halting, the node becomes a leaf
  
  - The leaf may hold the most frequent class among the subset tuples

- **Problem**
  - Difficult to choose an appropriate threshold
Tree Pruning Approaches

- **2nd approach: postpruning**
  - Remove branches from a “fully grown” tree—get a sequence of progressively pruned trees
  - A subtree at a given node is pruned by replacing it by a leaf
  - The leaf is labeled with the most frequent class

- **Example: cost complexity pruning algorithm**
  - Cost complexity of a tree is a function of the number of leaves and the error rate (percentage of tuples misclassified by the tree)
  - At each node $N$ compute
    - The cost complexity of the subtree at $N$
    - The cost complexity of the subtree at $N$ if it were to be pruned
  - If pruning results in smaller cost, then prune the subtree at $N$
  - Use a set of data different from the training data to decide which is the “best pruned tree”
2.2.4 Scalability and Decision Tree Induction

For scalable classification, propose presorting techniques on disk-resident data sets that are too large to fit in memory.

- **SLIQ** (EDBT’96 — Mehta et al.)
  - Builds an index for each attribute and only class list and the current attribute list reside in memory

- **SPRINT** (VLDB’96 — J. Shafer et al.)
  - Constructs an attribute list data structure

- **PUBLIC** (VLDB’98 — Rastogi & Shim)
  - Integrates tree splitting and tree pruning: stop growing the tree earlier

- **RainForest** (VLDB’98 — Gehrke, Ramakrishnan & Ganti)
  - Builds an AVC-list (attribute, value, class label)

- **BOAT** (PODS’99 — Gehrke, Ganti, Ramakrishnan & Loh)
  - Uses bootstrapping to create several small samples
Summary of Section 2.2

- Decision Trees have relatively **faster learning** speed than other methods
- Conversable to simple and **easy to understand** classification rules
- Information Gain, Ratio Gain and Gini Index are the most common methods of **attribute selection**
- **Tree pruning** is necessary to remove unreliable branches
- **Scalability** is an issue for large datasets
Chapter 2: Classification & Prediction

- 2.1 Basic Concepts of Classification and Prediction
- 2.2 Decision Tree Induction
  - 2.2.1 The Algorithm
  - 2.2.2 Attribute Selection Measures
  - 2.2.3 Tree Pruning
  - 2.2.4 Scalability and Decision Tree Induction
- 2.3 Bayes Classification Methods
  - 2.3.1 Naïve Bayesian Classification
  - 2.3.2 Note on Bayesian Belief Networks
- 2.4 Rule Based Classification
- 2.5 Lazy Learners
- 2.6 Prediction
- 2.7 How to Evaluate and Improve Classification
2.3 Bayes Classification Methods

- **What are Bayesian Classifiers?**
  - Statistical classifiers
  - Predict class membership probabilities: probability of a given tuple belonging to a particular class
  - Based on Bayes’ Theorem

- **Characteristics?**
  - Comparable performance with decision tree and selected neural network classifiers

- **Bayesian Classifiers**
  - Naïve Bayesian Classifiers
    - Assume independency between the effect of a given attribute on a given class and the other values of other attributes
  - Bayesian Belief Networks
    - Graphical models
    - Allow the representation of dependencies among subsets of attributes
Bayes’ Theorem In the Classification Context

- $X$ is a data tuple. In Bayesian term it is considered “evidence”
- $H$ is some hypothesis that $X$ belongs to a specified class $C$

$$P(H \mid X) = \frac{P(X \mid H)P(H)}{P(X)}$$

- $P(H \mid X)$ is the posterior probability of $H$ conditioned on $X$

**Example:** predict whether a costumer will buy a computer or not
- Costumers are described by two attributes: age and income
- $X$ is a 35 years-old costumer with an income of 40k
- $H$ is the hypothesis that the costumer will buy a computer
- $P(H \mid X)$ reflects the probability that costumer $X$ will buy a computer given that we know the costumers’ age and income
Bayes’ Theorem In the Classification Context

- **X** is a data tuple. In Bayesian term it is considered “evidence”
- **H** is some **hypothesis** that **X** belongs to a specified class **C**

\[
P(H | X) = \frac{P(X | H) P(H)}{P(X)}
\]

- **P(X| H)** is the posterior probability of **X** conditioned on **H**

**Example:** predict whether a customer will buy a computer or not
- Customers are described by two attributes: **age** and **income**
- **X** is a 35 years-old customer with an income of 40k
- **H** is the hypothesis that the customer will buy a computer
- **P(X| H)** reflects the probability that customer **X**, is 35 years-old and earns 40k, given that we know that the customer will buy a computer
Bayes’ Theorem In the Classification Context

- **X** is a data tuple. In Bayesian term it is considered “evidence”
- **H** is some hypothesis that X belongs to a specified class C

\[
P(H | X) = \frac{P(X | H) P(H)}{P(X)}
\]

- **P(H)** is the prior probability of **H**

**Example:** predict whether a customer will buy a computer or not

→ **H** is the hypothesis that the customer will buy a computer
→ The prior probability of **H** is the probability that a customer will buy a computer, regardless of age, income, or any other information for that matter
→ The posterior probability **P(H|X)** is based on more information than the prior probability **P(H)** which is independent from X
Bayes’ Theorem In the Classification Context

- **X** is a data tuple. In Bayesian term it is considered “evidence”
- **H** is some **hypothesis** that X belongs to a specified class C

\[
P(H \mid X) = \frac{P(X \mid H)P(H)}{P(X)}
\]

- **P(X)** is the **prior** probability of **X**
  - **Example:** predict whether a customer **will buy a computer or not**
    - Costumers are described by two attributes: **age** and **income**
    - **X** is a 35 years-old costumer with an income of 40k
    - **P(X)** is the probability that a person from our set of costumers is 35 years-old and earns 40k
**Naïve Bayesian Classification**

**D:** A training set of tuples and their associated class labels
Each tuple is represented by n-dimensional vector \( X(x_1,\ldots,x_n) \), n measurements of n attributes \( A_1,\ldots,A_n \)

**Classes:** suppose there are \( m \) classes \( C_1,\ldots,C_m \)

**Principle**

- Given a tuple \( X \), the classifier will predict that \( X \) belongs to the class having the **highest posterior probability** conditioned on \( X \)
- Predict that tuple \( X \) belongs to the class \( C_i \) if and only if

\[
P(C_i \mid X) > P(C_j \mid X) \quad \text{for} \quad 1 \leq j \leq m, \ j \neq i
\]

- Maximize \( P(C_i \mid X) \): find the **maximum posteriori hypothesis**

\[
P(C_i \mid X) = \frac{P(X \mid C_i)P(C_i)}{P(X)}
\]

- \( P(X) \) is **constant** for all classes, thus, **maximize** \( P(X \mid C_i)P(C_i) \)
Naïve Bayesian Classification

- To maximize $P(X|C_i)P(C_i)$, we need to know class prior probabilities
  - If the probabilities are not known, assume that $P(C_1)=P(C_2)=\ldots=P(C_m) \Rightarrow$ maximize $P(X|C_i)$
  - Class prior probabilities can be estimated by $P(C_i) = \frac{|C_i,D|}{|D|}$
- Assume **Class Conditional Independence** to reduce computational cost of $P(X|C_i)$
  - given $X(x_1,\ldots,x_n)$, $P(X|C_i)$ is:
    \[
P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i)
    \]
    \[
    = P(x_1|C_i) \times P(x_2|C_i) \times \ldots \times P(x_n|C_i)
    \]
  - The probabilities $P(x_1|C_i), \ldots P(x_n|C_i)$ can be estimated from the training tuples
Estimating $P(x_i | C_i)$

- **Categorical Attributes**
  - Recall that $x_k$ refers to the value of attribute $A_k$ for tuple $X$
  - $X$ is of the form $X(x_1, \ldots, x_n)$
  - $P(x_k | C_i)$ is the number of tuples of class $C_i$ in $D$ having the value $x_k$ for $A_k$, divided by $|C_{i,D}|$, the number of tuples of class $C_i$ in $D$
  - **Example**
    - 8 customers in class $C_{yes}$ (customer will buy a computer)
    - 3 customers among the 8 customers have high income
    - $P(\text{income}=\text{high} | C_{yes})$ the probability of a customer having a high income knowing that he belongs to class $C_{yes}$ is $3/8$

- **Continuous-Valued Attributes**
  - A continuous-valued attribute is assumed to have a Gaussian (Normal) distribution with mean $\mu$ and standard deviation $\sigma$

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(x-\mu)^2}{2\sigma^2}}$$
Estimating $P(x_i | C_i)$

- **Continuous-Valued Attributes**
  - The probability $P(x_k | C_i)$ is given by:
    \[
P(x_k | C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})
    \]
  - Estimate $\mu_{C_i}$ and $\sigma_{C_i}$ the mean and standard variation of the values of attribute $A_k$ for training tuples of class $C_i$
  - **Example**
    - $X$ a 35 years-old costumer with an income of 40k (age, income)
    - Assume the age attribute is continuous-valued
    - Consider class $C_{yes}$ (the costumer will buy a computer)
    - We find that in $D$, the costumers who will buy a computer are $38 \pm 12$ years of age $\Rightarrow \mu_{C_{yes}} = 38$ and $\sigma_{C_{yes}} = 12$

\[
P(\text{age} = 35 | C_{yes}) = g(35, 38, 12)
\]
<table>
<thead>
<tr>
<th>RID</th>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit-rating</th>
<th>class: buy_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>youth</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>youth</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td>middle-aged</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>senior</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>5</td>
<td>senior</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>6</td>
<td>senior</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>7</td>
<td>middle-aged</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>8</td>
<td>youth</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>9</td>
<td>youth</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>10</td>
<td>senior</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>11</td>
<td>youth</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>12</td>
<td>middle-aged</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>13</td>
<td>middle-aged</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>14</td>
<td>senior</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
</tbody>
</table>

Tuple to classify is

\[ X (\text{age}=\text{youth}, \text{income}=\text{medium}, \text{student}=\text{yes}, \text{credit}=\text{fair}) \]

Maximize \( P(X|C_i)P(C_i) \), for \( i=1,2 \)
Example

Given $X (\text{age}=\text{youth}, \text{income}=\text{medium}, \text{student}=\text{yes}, \text{credit}=\text{fair})$

Maximize $P(X|C_i)P(C_i)$, for $i=1,2$

**First step**: Compute $P(C_i)$. The prior probability of each class can be computed based on the training tuples:

- $P(\text{buys_computer}=\text{yes})=9/14=0.643$
- $P(\text{buys_computer}=\text{no})=5/14=0.357$

**Second step**: Compute $P(X|C_i)$ using the following conditional prob.

- $P(\text{age}=\text{youth}| \text{buys_computer}=\text{yes})=0.222$
- $P(\text{age}=\text{youth}| \text{buys_computer}=\text{no})=3/5=0.666$
- $P(\text{income}=\text{medium}| \text{buys_computer}=\text{yes})=0.444$
- $P(\text{income}=\text{medium}| \text{buys_computer}=\text{no})=2/5=0.400$
- $P(\text{student}=\text{yes}| \text{buys_computer}=\text{yes})=6/9=0.667$
- $P(\text{student}=\text{yes}| \text{buys_computer}=\text{no})=1/5=0.200$
- $P(\text{credit_rating}=\text{fair}| \text{buys_computer}=\text{yes})=6/9=0.667$
- $P(\text{credit_rating}=\text{fair}| \text{buys_computer}=\text{no})=2/5=0.400$
Example

\[ P(X| \text{buys\_computer=\text{yes}}) = P(\text{age=\text{youth}}| \text{buys\_computer=\text{yes}}) \times P(\text{income=\text{medium}}| \text{buys\_computer=\text{yes}}) \times P(\text{student=\text{yes}}| \text{buys\_computer=\text{yes}}) \times P(\text{credit\_rating=\text{fair}}| \text{buys\_computer=\text{yes}}) = 0.044 \]

\[ P(X| \text{buys\_computer=\text{no}}) = P(\text{age=\text{youth}}| \text{buys\_computer=\text{no}}) \times P(\text{income=\text{medium}}| \text{buys\_computer=\text{no}}) \times P(\text{student=\text{yes}}| \text{buys\_computer=\text{no}}) \times P(\text{credit\_rating=\text{fair}}| \text{buys\_computer=\text{no}}) = 0.019 \]

**Third step:** compute \( P(X| C_i)P(C_i) \) for each class

\[ P(X| \text{buys\_computer=\text{yes}})P(\text{buys\_computer=\text{yes}}) = 0.044 \times 0.643 = 0.028 \]

\[ P(X| \text{buys\_computer=\text{no}})P(\text{buys\_computer=\text{no}}) = 0.019 \times 0.357 = 0.007 \]

The naïve Bayesian Classifier predicts \( \text{buys\_computer=\text{yes}} \) for tuple \( X \)
Avoiding the 0-Probability Problem

- Naïve Bayesian prediction requires each conditional prob. be non-zero. Otherwise, the predicted prob. will be zero

\[
P(X | C_i) = \prod_{k=1}^{n} P(x_k | C_i)
\]

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income=medium (990), and income=high (10),

- **Use Laplacian correction** (or Laplacian estimator)
  - Adding 1 to each case
    - Prob(income = low) = 1/1003
    - Prob(income = medium) = 991/1003
    - Prob(income = high) = 11/1003
  - The “corrected” prob. estimates are close to their “uncorrected” counterparts
Summary of Section 2.3

- **Advantages**
  - Easy to implement
  - Good results obtained in most of the cases

- **Disadvantages**
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.
    - Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayesian Classifier

- **How to deal with these dependencies?**
  - Bayesian Belief Networks
2.3.2 Bayesian Belief Networks

- Bayesian belief network allows a subset of the variables conditionally independent
- A graphical model of causal relationships
  - Represents dependency among the variables
  - Gives a specification of joint probability distribution

![Diagram of Bayesian Belief Network]

→ Nodes: random variables
→ Links: dependency
→ X and Y are the parents of Z, and Y is the parent of P
→ No dependency between Z and P
→ Has no loops or cycles
The **conditional probability table** (CPT) for variable LungCancer:

<table>
<thead>
<tr>
<th></th>
<th>(FH, S)</th>
<th>(FH, ~S)</th>
<th>(~FH, S)</th>
<th>(~FH, ~S)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LC</strong></td>
<td>0.8</td>
<td>0.5</td>
<td>0.7</td>
<td>0.1</td>
</tr>
<tr>
<td>~<strong>LC</strong></td>
<td>0.2</td>
<td>0.5</td>
<td>0.3</td>
<td>0.9</td>
</tr>
</tbody>
</table>

CPT shows the conditional probability for each possible combination of its parents.

Derivation of the probability of a particular combination of values of X, from CPT:

\[
P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid \text{Parents (} Y_i \text{)})
\]
Several scenarios:

- Given both the network structure and all variables observable: learn only the CPTs.

- Network structure known, some hidden variables: gradient descent (greedy hill-climbing) method, analogous to neural network learning.

- Network structure unknown, all variables observable: search through the model space to reconstruct network topology.

- Unknown structure, all hidden variables: No good algorithms known for this purpose.
Bayesian Classifiers are statistical classifiers. They provide good accuracy. Naïve Bayesian classifier assumes independence between attributes. Causal relations are captured by Bayesian Belief Networks.