Notes

To preview next lecture:
- Check the lecture notes, if slides are not available:
  http://web.cse.ohio-state.edu/~sun.397/courses/au2017/cse5243-new.html
- Check UIUC course on the same topic. All their slides are available:
  https://wiki.illinois.edu/wiki/display/cs412/2.+Course+Syllabus+and+Schedule

Extra readings beyond the lecture slides are important for related job hunting or research:
- Lecture notes (or, chapters from the text book Han et al.)
Classification (Basic Concepts)
Huan Sun, CSE@The Ohio State University
Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary
Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set
Supervised vs. Unsupervised Learning

- **Supervised learning (classification)**
  - Supervision: The training data (observations, measurements, etc.) are accompanied by **labels** indicating the class of the observations
  - New data is classified based on the training set

- **Unsupervised learning (clustering)**
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data
Prediction Problems: Classification vs. Numeric Prediction

- **Classification**
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data

- **Numeric Prediction**
  - models continuous-valued functions, i.e., predicts unknown or missing values
Prediction Problems: Classification vs. Numeric Prediction

- **Classification**
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data

- **Numeric Prediction**
  - models continuous-valued functions, i.e., predicts unknown or missing values

- **Typical applications**
  - Credit/loan approval:
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is
Classification—A Two-Step Process

(1) **Model construction:** describing a set of predetermined classes

- Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
- The set of tuples used for model construction is **training set**
- Model: represented as classification rules, decision trees, or mathematical formulae
Classification—A Two-Step Process

(1) **Model construction:** describing a set of predetermined classes
   - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute.
   - The set of tuples used for model construction is **training set**.
   - Model: represented as classification rules, decision trees, or mathematical formulae.

(2) **Model usage:** for classifying future or unknown objects
   - Estimate accuracy of the model.
     - The known label of test sample is compared with the classified result from the model.
     - **Accuracy:** % of test set samples that are correctly classified by the model.
     - Test set is independent of training set (otherwise overfitting).
   - If the accuracy is acceptable, use the model to classify new data.
Classification—A Two-Step Process

(1) **Model construction:** describing a set of predetermined classes
- Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
- The set of tuples used for model construction is training set
- Model: represented as classification rules, decision trees, or mathematical formulae

(2) **Model usage:** for classifying future or unknown objects
- Estimate accuracy of the model
  - The known label of test sample is compared with the classified result from the model
  - **Accuracy:** % of test set samples that are correctly classified by the model
  - Test set is independent of training set (otherwise overfitting)
- If the accuracy is acceptable, use the model to classify new data

- Note: If the test set is used to select/refine models, it is called **validation (test) set** or development test set
Step (1): Model Construction

**Training Data**

<table>
<thead>
<tr>
<th>NAME</th>
<th>RANK</th>
<th>YEARS</th>
<th>TENURED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mike</td>
<td>Assistant Prof</td>
<td>3</td>
<td>no</td>
</tr>
<tr>
<td>Mary</td>
<td>Assistant Prof</td>
<td>7</td>
<td>yes</td>
</tr>
<tr>
<td>Bill</td>
<td>Professor</td>
<td>2</td>
<td>yes</td>
</tr>
<tr>
<td>Jim</td>
<td>Associate Prof</td>
<td>7</td>
<td>yes</td>
</tr>
<tr>
<td>Dave</td>
<td>Assistant Prof</td>
<td>6</td>
<td>no</td>
</tr>
<tr>
<td>Anne</td>
<td>Associate Prof</td>
<td>3</td>
<td>no</td>
</tr>
</tbody>
</table>

**Classification Algorithms**

Classifier (Model)
Step (1): Model Construction

Training Data

<table>
<thead>
<tr>
<th>NAME</th>
<th>RANK</th>
<th>YEARS</th>
<th>TENURED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mike</td>
<td>Assistant Prof</td>
<td>3</td>
<td>no</td>
</tr>
<tr>
<td>Mary</td>
<td>Assistant Prof</td>
<td>7</td>
<td>yes</td>
</tr>
<tr>
<td>Bill</td>
<td>Professor</td>
<td>2</td>
<td>yes</td>
</tr>
<tr>
<td>Jim</td>
<td>Associate Prof</td>
<td>7</td>
<td>yes</td>
</tr>
<tr>
<td>Dave</td>
<td>Assistant Prof</td>
<td>6</td>
<td>no</td>
</tr>
<tr>
<td>Anne</td>
<td>Associate Prof</td>
<td>3</td>
<td>no</td>
</tr>
</tbody>
</table>

Classification Algorithms

IF rank = ‘professor’ OR years > 6 THEN tenured = ‘yes’
Step (2): Using the Model in Prediction

<table>
<thead>
<tr>
<th>NAME</th>
<th>RANK</th>
<th>YEARS</th>
<th>TENURED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tom</td>
<td>Assistant Prof</td>
<td>2</td>
<td>no</td>
</tr>
<tr>
<td>Merlisa</td>
<td>Associate Prof</td>
<td>7</td>
<td>no</td>
</tr>
<tr>
<td>George</td>
<td>Professor</td>
<td>5</td>
<td>yes</td>
</tr>
<tr>
<td>Joseph</td>
<td>Assistant Prof</td>
<td>7</td>
<td>yes</td>
</tr>
</tbody>
</table>
Step (2): Using the Model in Prediction

<table>
<thead>
<tr>
<th>NAME</th>
<th>RANK</th>
<th>YEARS</th>
<th>TENURED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tom</td>
<td>Assistant Prof</td>
<td>2</td>
<td>no</td>
</tr>
<tr>
<td>Merlisa</td>
<td>Associate Prof</td>
<td>7</td>
<td>no</td>
</tr>
<tr>
<td>George</td>
<td>Professor</td>
<td>5</td>
<td>yes</td>
</tr>
<tr>
<td>Joseph</td>
<td>Assistant Prof</td>
<td>7</td>
<td>yes</td>
</tr>
</tbody>
</table>

Testing Data

Classifier

New/Unseen Data

(Jeff, Professor, 4)

Tenured? Yes
Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary
Decision Tree Induction: An Example

- Training data set: Buys_computer
- The data set follows an example of Quinlan’s ID3 (Playing Tennis)
Decision Tree Induction: An Example

- Training data set: Buys_computer
- The data set follows an example of Quinlan’s ID3 (Playing Tennis)
- Resulting tree:

<table>
<thead>
<tr>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rating</th>
<th>buys_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
</tbody>
</table>
Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - 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 are 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)
Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - 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 are 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**)

- 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
Algorithm Outline

- Split (node, {data tuples})
  - A <= the best attribute for splitting the {data tuples}
  - Decision attribute for this node <= A
  - For each value of A, create new child node
  - For each child node / subset:
    - If one of the stopping conditions is satisfied: STOP
    - Else: Split (child_node, {subset})

ID3 algorithm: how it works
https://www.youtube.com/watch?v=QhOdSLIE5c
Algorithm Outline

- **Split (node, {data tuples})**
  - $A \leq \text{the best attribute for splitting the } \{\text{data tuples}\}$
  - Decision attribute for this node $\leq A$
  - For each value of $A$, create new child node
  - For each child node / subset:
    - If one of the stopping conditions is satisfied: STOP
    - Else: Split (child_node, {subset})
Brief Review of Entropy

- **Entropy (Information Theory)**
  - A measure of uncertainty associated with a random number
  - Calculation: For a discrete random variable $Y$ taking $m$ distinct values $\{y_1, y_2, \ldots, y_m\}$
    \[
    H(Y) = -\sum_{i=1}^{m} p_i \log(p_i) \quad \text{where } p_i = P(Y = y_i)
    \]
  - Interpretation
    - Higher entropy $\rightarrow$ higher uncertainty
    - Lower entropy $\rightarrow$ lower uncertainty

- **Conditional entropy
  \[
  H(Y|X) = \sum_x p(x) H(Y|X = x)
  \]
Attribute Selection Measure: Information Gain (ID3/C4.5)

- 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 $D$:
  \[
  \text{Info}_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \text{Info}(D_j)
  \]
- Information gained by branching on attribute $A$
  \[
  \text{Gain}(A) = \text{Info}(D) - \text{Info}_A(D)
  \]
Attribute Selection: Information Gain

- Class P: buys_computer = “yes”
- Class N: buys_computer = “no”

<table>
<thead>
<tr>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rating</th>
<th>buys_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
</tbody>
</table>

How to select the first attribute?
Attribute Selection: Information Gain

- Class P: `buys_computer = “yes”`
- Class N: `buys_computer = “no”`

\[
\text{Info}(D) = I(9,5) = -\frac{9}{14} \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \log_2\left(\frac{5}{14}\right) = 0.940
\]

<table>
<thead>
<tr>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rating</th>
<th>buys_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
</tbody>
</table>
Attribute Selection: Information Gain

- Class P: buys_computer = “yes”
- Class N: buys_computer = “no”

\[ \text{Info}(D) = I(9,5) = - \frac{9}{14} \log_2 \left( \frac{9}{14} \right) - \frac{5}{14} \log_2 \left( \frac{5}{14} \right) = 0.940 \]

Look at “age”:

<table>
<thead>
<tr>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rating</th>
<th>buys_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
</tbody>
</table>

\[
\begin{array}{ccc}
\text{age} & p_i & n_i & I(p_i, n_i) \\
<=30 & 2 & 3 & 0.971 \\
31...40 & 4 & 0 & 0 \\
>40 & 3 & 2 & 0.971 \\
\end{array}
\]
Attribute Selection: Information Gain

- Class P: buys_computer = “yes”
- Class N: buys_computer = “no”

\[\text{Info}(D) = I(9, 5) = -\frac{9}{14} \log_2 \left( \frac{9}{14} \right) - \frac{5}{14} \log_2 \left( \frac{5}{14} \right) = 0.940\]

Look at “age”:

\[\text{Info}_{age}(D) = \frac{5}{14} I(2, 3) + \frac{4}{14} I(4, 0) + \frac{5}{14} I(3, 2) = 0.694\]
Attribute Selection: Information Gain

- Class P: buys_computer = “yes”
- Class N: buys_computer = “no”

Look at “age”:

<table>
<thead>
<tr>
<th>age</th>
<th>p_i</th>
<th>n_i</th>
<th>I(p_i, n_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>2</td>
<td>3</td>
<td>0.971</td>
</tr>
<tr>
<td>31...40</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>&gt;40</td>
<td>3</td>
<td>2</td>
<td>0.971</td>
</tr>
</tbody>
</table>

\[ Info_{age}(D) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.694 \]

\[ \frac{5}{14} I(2,3) \] means “age <=30” has 5 out of 14 samples, with 2 yes’es and 3 no’s.
## Attribute Selection: Information Gain

- **Class P**: `buys_computer` = “yes”
- **Class N**: `buys_computer` = “no”

<table>
<thead>
<tr>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rating</th>
<th>buys_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
</tbody>
</table>

**Information Gain**

\[ \text{Info}(D) = I(9,5) = - \frac{9}{14} \log_2 \left( \frac{9}{14} \right) - \frac{5}{14} \log_2 \left( \frac{5}{14} \right) = 0.940 \]

\[ \text{Info}_{age}(D) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.694 \]

**Gain(age)**

\[ \text{Gain}(age) = \text{Info}(D) - \text{Info}_{age}(D) = 0.246 \]

Similarly,

\[ \text{Gain}(income) = 0.029 \]

\[ \text{Gain}(student) = 0.151 \]

\[ \text{Gain}(credit\_rating) = 0.048 \]
Recursive Procedure

<table>
<thead>
<tr>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rating</th>
<th>buys_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
</tbody>
</table>

1. After selecting age at the root node, we will create three child nodes.

2. One child node is associated with red data tuples.

3. How to continue for this child node?
Recursive Procedure

1. After selecting age at the root node, we will create three child nodes.

2. One child node is associated with red data tuples.

3. How to continue for this child node?

   Now, you will make $D = \{\text{red data tuples}\}$ and then select the best attribute to further split $D$.

   A recursive procedure.
Let attribute A be a continuous-valued attribute
Let attribute A be a continuous-valued attribute

Must determine the best split point for A (or, discretization)

=>

D1 is the set of tuples in D satisfying $A \leq \text{split-point}$, and D2 is the set of tuples in D satisfying $A > \text{split-point}$
Computing Information-Gain for Continuous-Valued Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the best split point for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible split point
    - \( \frac{a_i + a_{i+1}}{2} \) is the midpoint between the values of \( a_i \) and \( a_{i+1} \)
  - The point with the minimum expected information requirement for A (i.e., \( \text{Info}_A(D) \)) is selected as the split-point for A
Computing Information-Gain for Continuous-Valued Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the best split point for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible split point
    - \((a_i + a_{i+1})/2\) is the midpoint between the values of \(a_i\) and \(a_{i+1}\)
  - The point with the minimum expected information requirement for A (i.e., \(\text{Info}_A(D)\)) is selected as the split-point for A
- Split:
  - \(D_1\) is the set of tuples in D satisfying \(A \leq \text{split-point}\), and \(D_2\) is the set of tuples in D satisfying \(A > \text{split-point}\)
How to Select Test Attribute?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous

- Depends on number of ways to split
  - 2-way split
  - Multi-way split
Splitting Based on Nominal Attributes

- **Multi-way split**: Use as many partitions as distinct values.

  - CarType
    - Family
    - Luxury
    - Sports

- **Binary split**: Divides values into two subsets. Need to find optimal partitioning.

  - OR
    - CarType
      - {Sports, Luxury}
      - {Family}
    - OR
      - CarType
        - {Family, Luxury}
        - {Sports}
Splitting Based on Continuous Attributes

(i) Binary split

(ii) Multi-way split
How to Determine the Best Split

- **Greedy approach:**
  - Nodes with *homogeneous* class distribution are preferred

  Ideally, data tuples at that node belong to the same class.

<table>
<thead>
<tr>
<th>C0: 5</th>
<th>C0: 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1: 5</td>
<td>C1: 1</td>
</tr>
</tbody>
</table>

Non-homogeneous, High degree of impurity

Homogeneous, Low degree of impurity
Rethink about Decision Tree Classification

- **Greedy approach:**
  - Nodes with *homogeneous* class distribution are preferred

- **Need a measure of node impurity:**

<table>
<thead>
<tr>
<th>C0: 5</th>
<th>C0: 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1: 5</td>
<td>C1: 1</td>
</tr>
</tbody>
</table>

  - Non-homogeneous, High degree of impurity
  - Homogeneous, Low degree of impurity
Measures of Node Impurity

- **Entropy:**
  \[ H(Y) = -\sum_{i=1}^{m} p_i \log(p_i) \text{ where } p_i = P(Y = y_i) \]

  - Higher entropy \(\Rightarrow\) higher uncertainty, higher node impurity
  - Why entropy is used in information gain

- **Gini Index**

- **Misclassification error**
Potential Problems with Information Gain

- Information gain measure is biased towards attributes with a large number of values (or, a large number of small partitions)
Information gain measure is biased towards attributes with a large number of values (or, a large number of small partitions)

Consider an attribute that acts as a unique identifier, such as student_ID
Potential Problems with Information Gain

- Information gain measure is biased towards attributes with a large number of values (or, a large number of small partitions)
  
  - Consider an attribute that acts as a unique identifier, such as student_ID
  
  - A split on Student_ID will result in many partitions, each one containing just one tuple.
Potential Problems with Information Gain

- Information gain measure is biased towards attributes with a large number of values (or, a large number of small partitions)
  - Consider an attribute that acts as a unique identifier, such as student_ID
  - A split on Student_ID will result in many partitions, each one containing just one tuple.
  - Information required to classify data set D after this partitioning is:

$$\text{Info}_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \text{Info}(D_j)$$
Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)
Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values.
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

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

- The entropy of the partitioning, or the potential information generated by splitting \( D \) into \( v \) partitions.
Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

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

- The entropy of the partitioning, or the potential information generated by splitting \( D \) into \( v \) partitions.

- \( \text{GainRatio}(A) = \text{Gain}(A)/\text{SplitInfo}(A) \) (normalizing Information Gain)
Gain Ratio for Attribute Selection (C4.5)

- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

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

- GainRatio(A) = Gain(A)/SplitInfo(A)

- Ex.

<table>
<thead>
<tr>
<th>Gain(income) = 0.029</th>
<th>(from last class, slide 27)</th>
</tr>
</thead>
</table>

\[
SplitInfo_{\text{income}}(D) = -\frac{4}{14} \times \log_2 \left( \frac{4}{14} \right) - \frac{6}{14} \times \log_2 \left( \frac{6}{14} \right) - \frac{4}{14} \times \log_2 \left( \frac{4}{14} \right) = 1.557
\]

- gain\_ratio(income) = 0.029/1.557 = 0.019

- The attribute with the maximum gain ratio is selected as the splitting attribute
Another Attribute Selection Measure: Gini Index (CART, IBM IntelligentMiner)

- If a data set $D$ contains examples from $n$ classes, gini index, $gini(D)$ is defined as

$$gini(D) = 1 - \sum_{j=1}^{n} P_j^2$$

where $p_j$ is the relative frequency of class $j$ in $D$.

Gini index measures the impurity of $D$. The larger, the more impure.
Another Attribute Selection Measure: Gini Index (CART, IBM IntelligentMiner)

- If a data set $D$ contains examples from $n$ classes, gini index, $gini(D)$ is defined as

$$gini(D) = 1 - \sum_{j=1}^{n} P_j^2$$

where $p_j$ is the relative frequency of class $j$ in $D$

Ex. D has 9 tuples in buys_computer = “yes” and 5 in “no”

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$
Gini Index (CART, IBM IntelligentMiner)

- If a data set $D$ contains examples from $n$ classes, gini index, $gini(D)$ is defined as
  \[
gini(D) = 1 - \sum_{j=1}^{n} p_j^2,
\]
  where $p_j$ is the relative frequency of class $j$ in $D$.

- If a data set $D$ is split on $A$ into two subsets $D_1$ and $D_2$, the gini index after the split is defined as
  \[
gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2).
\]
Gini Index (CART, IBM IntelligentMiner)

- If a data set \( D \) contains examples from \( n \) classes, gini index, \( gini(D) \) is defined as
  \[
gini(D) = 1 - \sum_{j=1}^{n} p_j^2,
\]
  where \( p_j \) is the relative frequency of class \( j \) in \( D \).

- If a data set \( D \) is split on \( A \) into two subsets \( D_1 \) and \( D_2 \), the gini index \( gini(D) \) is defined as
  \[
gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)
\]

- Reduction in impurity:
  \[
  \Delta gini(A) = gini(D) - gini_A(D)
\]
Gini Index (CART, IBM IntelligentMiner)

- If a data set $D$ contains examples from $n$ classes, gini index, $gini(D)$ is defined as
  \[ gini(D) = 1 - \sum_{j=1}^{n} p_j^2 \]
  where $p_j$ is the relative frequency of class $j$ in $D$

- If a data set $D$ is split on $A$ into two subsets $D_1$ and $D_2$, the gini index after the split is defined as
  \[ gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2) \]

- Reduction in impurity:
  \[ \Delta gini(A) = gini(D) - gini_A(D) \]

- The attribute provides the smallest $gini_A(D)$ (or, the largest reduction in impurity) is chosen to split the node.
Gini Index (CART, IBM IntelligentMiner)

- If a data set $D$ contains examples from $n$ classes, gini index, $gini(D)$ is defined as
  \[
gini(D) = 1 - \sum_{j=1}^{n} p_j^2,
\]
  where $p_j$ is the relative frequency of class $j$ in $D$.

- If a data set $D$ is split on $A$ into two subsets $D_1$ and $D_2$, the gini index after the split is defined as
  \[
gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2).
\]

- Reduction in impurity:
  \[
  \Delta gini(A) = gini(D) - gini_A(D)
\]

- The attribute provides the smallest $gini_A(D)$ (or, the largest reduction in impurity) is chosen to split the node.
Binary Attributes: Computing Gini Index

- Splits into two partitions
- Effect of weighing partitions:
  - Larger and Purer Partitions are sought for.

\[
gini(D) = 1 - \sum_{j=1}^{n} p_j^2
\]

<table>
<thead>
<tr>
<th>Parent</th>
<th>Gini = ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>6</td>
</tr>
<tr>
<td>C2</td>
<td>6</td>
</tr>
</tbody>
</table>
Binary Attributes: Computing Gini Index

- Splits into two partitions
- Effect of weighing partitions:
  - Larger and Purer Partitions are sought for.

\[
gini(D) = 1 - \sum_{j=1}^{n} p_j^2
\]

<table>
<thead>
<tr>
<th></th>
<th>Parent</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>6</td>
</tr>
<tr>
<td>C2</td>
<td>6</td>
</tr>
</tbody>
</table>

Gini = 0.500

\[
\text{Gini}(N1) = 1 - \frac{5}{6}^2 - \frac{2}{6}^2 = 0.194
\]

\[
\text{Gini}(N2) = 1 - \frac{1}{6}^2 - \frac{4}{6}^2 = 0.528
\]

\[
\begin{array}{c|c|c}
 & \text{N1} & \text{N2} \\
\hline
\text{C1} & 5 & 1 \\
\text{C2} & 2 & 4 \\
\end{array}
\]
Binary Attributes: Computing Gini Index

- Splits into two partitions
- Effect of weighing partitions:
  - Larger and Purer Partitions are sought for.

\[
gini(D) = 1 - \sum_{j=1}^{n} p_j^2
\]

\[
\begin{array}{c|c|c|c}
 & 
\text{Node N1} & 
\text{Node N2} & 
\text{Gini} \\
\hline
\text{C1} & 5 & 1 & 0.194 \\
\text{C2} & 2 & 4 & 0.528 \\
\end{array}
\]

\[
\text{Gini(Children)} = \frac{7}{12} \times 0.194 + \frac{5}{12} \times 0.528 = 0.333
\]

\[
\text{Gini(N1)} = 1 - (\frac{5}{6})^2 - (\frac{2}{6})^2 = 0.194
\]

\[
\text{Gini(N2)} = 1 - (\frac{1}{6})^2 - (\frac{4}{6})^2 = 0.528
\]

\[
\text{Gini(N1)} = \frac{7}{12} \times 0.194 + \frac{5}{12} \times 0.528 = 0.333
\]
Categorical Attributes: Computing Gini Index

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

### Multi-way split

<table>
<thead>
<tr>
<th>CarType</th>
<th>Family</th>
<th>Sports</th>
<th>Luxury</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Gini 0.393

### Two-way split (find best partition of values)

<table>
<thead>
<tr>
<th>CarType</th>
<th>{Sports, Luxury}</th>
<th>{Family}</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

Gini 0.400

<table>
<thead>
<tr>
<th>CarType</th>
<th>{Sports}</th>
<th>{Family, Luxury}</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>C2</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>

Gini 0.419
Continuous Attributes: Computing Gini Index

- Use Binary Decisions based on one value

- Several Choices for the splitting value
  - Number of possible splitting values = Number of distinct values - 1
  - Typically, the midpoint between each pair of adjacent values is considered as a possible split point
    - \((a_i + a_{i+1})/2\) is the midpoint between the values of \(a_i\) and \(a_{i+1}\)

- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions, \(A < v\) and \(A \geq v\)

- Simple method to choose best \(v\)
  - For each \(v\), scan the database to gather count matrix and compute its Gini index
  - Computationally Inefficient! Repetition of work.

```
<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Continuous Attributes: Computing Gini Index...

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least Gini index

<table>
<thead>
<tr>
<th>Cheat</th>
<th>No</th>
<th>No</th>
<th>No</th>
<th>Yes</th>
<th>Yes</th>
<th>Yes</th>
<th>No</th>
<th>No</th>
<th>No</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taxable Income</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>70</td>
<td>75</td>
<td>85</td>
<td>90</td>
<td>95</td>
<td>100</td>
<td>120</td>
<td>125</td>
<td>220</td>
<td></td>
</tr>
<tr>
<td>&lt;=</td>
<td>&gt;</td>
<td>&lt;=</td>
<td>&gt;</td>
<td>&lt;=</td>
<td>&gt;</td>
<td>&lt;=</td>
<td>&gt;</td>
<td>&lt;=</td>
<td>&gt;</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cheat</th>
<th>No</th>
<th>No</th>
<th>No</th>
<th>Yes</th>
<th>Yes</th>
<th>Yes</th>
<th>No</th>
<th>No</th>
<th>No</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Split Positions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>72</td>
<td>80</td>
<td>87</td>
<td>92</td>
<td>97</td>
<td>110</td>
<td>122</td>
<td>172</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;=</td>
<td>&gt;</td>
<td>&lt;=</td>
<td>&gt;</td>
<td>&lt;=</td>
<td>&gt;</td>
<td>&lt;=</td>
<td>&gt;</td>
<td>&lt;=</td>
<td>&gt;</td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>72</td>
<td>80</td>
<td>87</td>
<td>92</td>
<td>97</td>
<td>110</td>
<td>122</td>
<td>172</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cheat</th>
<th>No</th>
<th>No</th>
<th>No</th>
<th>Yes</th>
<th>Yes</th>
<th>Yes</th>
<th>No</th>
<th>No</th>
<th>No</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gini</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.400</td>
<td>0.375</td>
<td>0.343</td>
<td>0.417</td>
<td>0.400</td>
<td>0.300</td>
<td>0.343</td>
<td>0.375</td>
<td>0.400</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Attribute Selection: Complexity Note

- Class P: buys_computer = “yes”
- Class N: buys_computer = “no”

<table>
<thead>
<tr>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rating</th>
<th>buys_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
</tbody>
</table>

Look at “age”:

<table>
<thead>
<tr>
<th>age</th>
<th>p_i</th>
<th>n_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>31…40</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>&gt;40</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

When selecting an attribute, scan the data set at each node to gather count matrix.
Another Impurity Measure: Misclassification Error

- Classification error at a node $t$:
  \[
  Error(t) = 1 - \max_i P(i \mid t)
  \]

- $P(i \mid t)$ means the relative frequency of class $i$ at node $t$.

- Measures misclassification error made by a node.
  - Maximum $(1 - 1/n_c)$ when records are equally distributed among all classes, implying most impurity
  - Minimum $(0.0)$ when all records belong to one class, implying least impurity
Examples for Misclassification Error

\[ Error(t) = 1 - \max_i P(i \mid t) \]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>P(C1)</th>
<th>P(C2)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>0</td>
<td>0/6 = 0</td>
<td>6/6 = 1</td>
<td>1 - max(0, 1) = 1 - 1 = 0</td>
</tr>
<tr>
<td>C2</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>P(C1)</th>
<th>P(C2)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1</td>
<td>1/6</td>
<td>5/6</td>
<td>1 - max(1/6, 5/6) = 1 - 5/6 = 1/6</td>
</tr>
<tr>
<td>C2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>P(C1)</th>
<th>P(C2)</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>2</td>
<td>2/6</td>
<td>4/6</td>
<td>1 - max(2/6, 4/6) = 1 - 4/6 = 1/3</td>
</tr>
<tr>
<td>C2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Comparison among Impurity Measure

For a 2-class problem:
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 others
  - **Gini index:**
    - biased to multivalued attributes
    - has difficulty when # of classes is large
    - tends to favor tests that result in equal-sized partitions and purity in both partitions
Other Attribute Selection Measures

- **CHAID**: a popular decision tree algorithm, measure based on $\chi^2$ test for independence
- **C-SEP**: performs better than info. gain and gini index in certain cases
- **G-statistic**: has a close approximation to $\chi^2$ distribution
- **MDL (Minimal Description Length) principle** (i.e., the simplest solution is preferred):
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- Multivariate splits (partition based on multiple variable combinations)
  - **CART**: finds multivariate splits based on a linear comb. of attrs.
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others
Decision Tree Based Classification

- Advantages:
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Accuracy is comparable to other classification techniques for many simple data sets
Example: C4.5

- Simple depth-first construction.
- Uses Information Gain
- Sorts Continuous Attributes at each node.
- Needs entire data to fit in memory.
- Unsuitable for Large Datasets.
  - Needs out-of-core sorting.

- You can download the software online, e.g.,
Practical Issues of Classification

- Underfitting and Overfitting
- Missing Values
- Costs of Classification
Underfitting and Overfitting

Underfitting: when model is too simple, both training and test errors are large

Overfitting

- When model is too complex
- High training accuracy but poor test accuracy
- Overfitting occurs when the model learns the noise in the training data

Graph:
- Error (%) vs. Number of nodes
- Red line: Training set
- Blue line with dots: Test set
- Overfitting occurs at around 200 nodes
Overfitting due to Noise

Decision boundary is distorted by noise point
Lack of data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region.

- Insufficient number of training records in the region causes the decision tree to predict the test examples using other training records that are irrelevant to the classification task.
Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary.
- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records.
- Need new ways for estimating errors.