Classification: Basic Concepts, Decision Trees, and Model Evaluation

Dr. Hui Xiong
Rutgers University

Classification: Definition

- Given a collection of records (training set)
  - Each record is characterized by a tuple \((x, y)\), where \(x\) is the attribute set and \(y\) is the class label
    - \(x\): attribute, predictor, independent variable, input
    - \(y\): class, response, dependent variable, output

- Task:
  - Learn a model that maps each attribute set \(x\) into one of the predefined class labels \(y\)
Examples of Classification Task

<table>
<thead>
<tr>
<th>Task</th>
<th>Attribute set, $x$</th>
<th>Class label, $y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Categorizing email messages</td>
<td>Features extracted from email message header and content</td>
<td>spam or non-spam</td>
</tr>
<tr>
<td>Identifying tumor cells</td>
<td>Features extracted from MRI scans</td>
<td>malignant or benign cells</td>
</tr>
<tr>
<td>Cataloging galaxies</td>
<td>Features extracted from telescope images</td>
<td>Elliptical, spiral, or irregular-shaped galaxies</td>
</tr>
</tbody>
</table>

General Approach for Building Classification Model

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrib1</th>
<th>Attrib2</th>
<th>Attrib3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No</td>
<td>Large</td>
<td>123K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Medium</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Small</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Medium</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Large</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Medium</td>
<td>65K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Large</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Small</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Medium</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Small</td>
<td>90K</td>
<td>?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrib1</th>
<th>Attrib2</th>
<th>Attrib3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>No</td>
<td>Small</td>
<td>55K</td>
<td>?</td>
</tr>
<tr>
<td>12</td>
<td>Yes</td>
<td>Medium</td>
<td>30K</td>
<td>?</td>
</tr>
<tr>
<td>13</td>
<td>Yes</td>
<td>Large</td>
<td>110K</td>
<td>?</td>
</tr>
<tr>
<td>14</td>
<td>No</td>
<td>Small</td>
<td>95K</td>
<td>?</td>
</tr>
<tr>
<td>15</td>
<td>No</td>
<td>Large</td>
<td>67K</td>
<td>?</td>
</tr>
</tbody>
</table>
Classification Techniques

- **Base Classifiers**
  - Decision Tree based Methods
  - Rule-based Methods
  - Nearest-neighbor
  - Neural Networks
  - Naïve Bayes and Bayesian Belief Networks
  - Support Vector Machines

- **Ensemble Classifiers**
  - Boosting, Bagging, Random Forests

### Example of a Decision Tree

<table>
<thead>
<tr>
<th>ID</th>
<th>Home Owner</th>
<th>Marital Status</th>
<th>Annual Income</th>
<th>Defaulted Borrower</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>
</tbody>
</table>

**Splitting Attributes**

- **Home Owner**: Yes -> NO, No -> MarSt
- **MarSt**: Single, Divorced -> Income
  - Income < 80K: NO
  - Income > 80K: YES
- **MarSt**: Married -> NO

Model: Decision Tree
### Another Example of Decision Tree

<table>
<thead>
<tr>
<th>ID</th>
<th>Home Owner</th>
<th>Marital Status</th>
<th>Annual Income</th>
<th>Defaulted</th>
<th>Borrower</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
<td>NO</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
<td>NO</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
<td>NO</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
<td>NO</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
<td>NO</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
<td>NO</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
<td>NO</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
<td>NO</td>
</tr>
</tbody>
</table>

There could be more than one tree that fits the same data!

---

### Decision Tree Classification Task

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attr1</th>
<th>Attr2</th>
<th>Attr3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Large</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Medium</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Small</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Medium</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Large</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Medium</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Large</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Small</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Medium</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Small</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attr1</th>
<th>Attr2</th>
<th>Attr3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>No</td>
<td>Small</td>
<td>55K</td>
<td>?</td>
</tr>
<tr>
<td>12</td>
<td>Yes</td>
<td>Medium</td>
<td>80K</td>
<td>?</td>
</tr>
<tr>
<td>13</td>
<td>Yes</td>
<td>Large</td>
<td>110K</td>
<td>?</td>
</tr>
<tr>
<td>14</td>
<td>No</td>
<td>Small</td>
<td>95K</td>
<td>?</td>
</tr>
<tr>
<td>15</td>
<td>No</td>
<td>Large</td>
<td>67K</td>
<td>?</td>
</tr>
</tbody>
</table>

Introduction to Data Mining 1/2/2009
Apply Model to Test Data

Start from the root of tree.

Test Data

<table>
<thead>
<tr>
<th>Home Owner</th>
<th>Marital Status</th>
<th>Annual Income</th>
<th>Defaulted Borrower</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

Apply Model to Test Data

Test Data

<table>
<thead>
<tr>
<th>Home Owner</th>
<th>Marital Status</th>
<th>Annual Income</th>
<th>Defaulted Borrower</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

Introduction to Data Mining 1/2/2009 9
Apply Model to Test Data

Test Data

<table>
<thead>
<tr>
<th>Home Owner</th>
<th>Marital Status</th>
<th>Annual Income</th>
<th>Defaulted Borrower</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

Diagram:
- Home Owner (No)
  - Income (No)
    - Single, Divorced
  - Income (YES)
    - > 80K
      - No
      - YES
  - Married

Introduction to Data Mining 1/2/2009
Apply Model to Test Data

Test Data

<table>
<thead>
<tr>
<th>Home Owner</th>
<th>Marital Status</th>
<th>Annual Income</th>
<th>Defaulted Borrower</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

Introduction to Data Mining 1/2/2009

Apply Model to Test Data

Test Data

<table>
<thead>
<tr>
<th>Home Owner</th>
<th>Marital Status</th>
<th>Annual Income</th>
<th>Defaulted Borrower</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

Assign Defaulted to "No"
**Decision Tree Classification Task**

![Decision Tree Induction Diagram](image)

**Decision Tree Induction**

- Many Algorithms:
  - Hunt’s Algorithm (one of the earliest)
  - CART
  - ID3, C4.5
  - SLIQ, SPRINT
General Structure of Hunt’s 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 $D_t$ 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.

---

Hunt’s Algorithm

(a) Defaulted = No
(b) Defaulted = Yes
(c) Defaulted = No
(d) Defaulted = Yes

Introduction to Data Mining 1/2/2009 17
Design Issues of Decision Tree Induction

- How should training records be split?
  - Method for specifying test condition
    - depending on attribute types
  - Measure for evaluating the goodness of a test condition

- How should the splitting procedure stop?
  - Stop splitting if all the records belong to the same class or have identical attribute values
  - Early termination

Methods for Expressing Test Conditions

- Depends on attribute types
  - Binary
  - Nominal
  - Ordinal
  - Continuous

- Depends on number of ways to split
  - 2-way split
  - Multi-way split
Test Condition for Nominal Attributes

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

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

Test Condition for Ordinal Attributes

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

- Binary split:
  - Divides values into two subsets
  - Need to find optimal partitioning
  - Preserve the order property among attribute values

This grouping violates order property
### Test Condition for Continuous Attributes

#### Annual Income $> 80K$?
- Yes
- No

#### Annual Income?
- $< 10K$
- $[10K,25K]$
- $[25K,50K]$
- $[50K,80K]$
- $> 80K$

(i) Binary split
(ii) Multi-way split

---

### Splitting Based on Continuous Attributes

- **Different ways of handling**
  - **Discretization** to form an ordinal categorical attribute
    - Static – discretize once at the beginning
    - Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.

- **Binary Decision**: $(A < v)$ or $(A \geq v)$
  - consider all possible splits and finds the best cut
  - can be more compute-intensive
How to determine the Best Split

Before Splitting: 10 records of class 0, 10 records of class 1

Which test condition is the best?

How to determine the Best Split

- Greedy approach:
  - Nodes with purer class distribution are preferred

- Need a measure of node impurity:

<table>
<thead>
<tr>
<th>C0: 5</th>
<th>C1: 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>High degree of impurity</td>
<td>Low degree of impurity</td>
</tr>
</tbody>
</table>

Introduction to Data Mining  1/2/2009  25
**Measures of Node Impurity**

- **Gini Index**
  \[
  \text{GINI}(t) = 1 - \sum_j [p(j \mid t)]^2
  \]

- **Entropy**
  \[
  \text{Entropy}(t) = -\sum_j p(j \mid t) \log p(j \mid t)
  \]

- **Misclassification error**
  \[
  \text{Error}(t) = 1 - \max P(i \mid t)
  \]

**Finding the Best Split**

1. Compute impurity measure (P) before splitting
2. Compute impurity measure (M) after splitting
   - Compute impurity measure of each child node
   - Compute the average impurity of the children (M)
3. Choose the attribute test condition that produces the highest gain
   \[
   \text{Gain} = P - M
   \]
   or equivalently, lowest impurity measure after splitting (M)
Finding the Best Split

Before Splitting:

\[ A? \]

Node N1

Yes

Node N2

No

\[ B? \]

Node N3

Yes

Node N4

No

\[ C0 \]

N00

P

\[ C1 \]

N01

\[ \text{Gain} = P - M1 \text{ vs } P - M2 \]

Gain = P - M1 vs P - M2

Measure of Impurity: GINI

- Gini Index for a given node \( t \):

\[
\text{GINI}(t) = 1 - \sum_j [p(j | t)]^2
\]

(Note: \( p(j | t) \) is the relative frequency of class \( j \) at node \( t \)).

- Maximum (1 - 1/\( n_c \)) when records are equally distributed among all classes, implying least interesting information
- Minimum (0.0) when all records belong to one class, implying most interesting information
Computing Gini Index of a Single Node

\[
GINI(t) = 1 - \sum_{j} [p(j | t)]^2
\]

<table>
<thead>
<tr>
<th>C1</th>
<th>0</th>
<th>P(C1) = 0/6 = 0     P(C2) = 6/6 = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>6</td>
<td>Gini = 1 – P(C1)^2 – P(C2)^2 = 1 – 0 – 1 = 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C1</th>
<th>1</th>
<th>P(C1) = 1/6       P(C2) = 5/6</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>5</td>
<td>Gini = 1 – (1/6)^2 – (5/6)^2 = 0.278</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C1</th>
<th>2</th>
<th>P(C1) = 2/6       P(C2) = 4/6</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>4</td>
<td>Gini = 1 – (2/6)^2 – (4/6)^2 = 0.444</td>
</tr>
</tbody>
</table>

Computing Gini Index for a Collection of Nodes

- When a node \( p \) is split into \( k \) partitions (children)

\[
GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)
\]

where, \( n_i \) = number of records at child \( i \),
\( n \) = number of records at parent node \( p \).

- Choose the attribute that minimizes weighted average Gini index of the children

- Gini index is used in decision tree algorithms such as CART, SLIQ, SPRINT
Binary Attributes: Computing Gini Index

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

\[
\text{Gini} = 1 - \left(\frac{5}{6}\right)^2 - \left(\frac{1}{6}\right)^2 = 0.278
\]

\[
\text{Gini}(N1) = 1 - \left(\frac{2}{6}\right)^2 - \left(\frac{4}{6}\right)^2 = 0.444
\]

\[
\text{Gini}(N2) = 1 - \left(\frac{2}{6}\right)^2 - \left(\frac{4}{6}\right)^2 = 0.444
\]

<table>
<thead>
<tr>
<th>Parent</th>
<th>N1</th>
<th>N2</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>C2</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

\[
\text{Gini(Children)} = 6/12 \times 0.278 + 6/12 \times 0.444 = 0.361
\]

Categorical Attributes: Computing Gini Index

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

\[
\begin{array}{c|c|c|c|c}
\text{CarType} & \text{Family} & \text{Sports} & \text{Luxury} \\
\hline
\text{C1} & 1 & 8 & 1 \\
\text{C2} & 3 & 0 & 7 \\
\text{Gini} & 0.163 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c}
\text{CarType} & \text{(Sports, Luxury)} & \text{Family} \\
\hline
\text{C1} & 9 & 1 \\
\text{C2} & 7 & 3 \\
\text{Gini} & 0.468 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c}
\text{CarType} & \text{(Sports)} & \text{(Family, Luxury)} \\
\hline
\text{C1} & 8 & 2 \\
\text{C2} & 0 & 10 \\
\text{Gini} & 0.167 \\
\end{array}
\]
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
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions, A < v and A ≥ 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.

For efficient computation:

- 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
Measure of Impurity: Entropy

- Entropy at a given node t:
  \[ Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t) \]

(NOTE: \(p(j \mid t)\) is the relative frequency of class j at node t).

- Maximum (log \(n_c\)) when records are equally distributed among all classes implying least information
- Minimum (0.0) when all records belong to one class, implying most information

Entropy based computations are quite similar to the GINI index computations

Computing Entropy of a Single Node

\[ Entropy(t) = -\sum_j p(j \mid t) \log_2 p(j \mid t) \]

<table>
<thead>
<tr>
<th>Class</th>
<th>Count</th>
<th>(P(C1))</th>
<th>(P(C2))</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>0</td>
<td>0/6 = 0</td>
<td>6/6 = 1</td>
<td>0</td>
</tr>
<tr>
<td>C2</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Entropy = \(-0 \log 0 - 1 \log 1 = -0 - 0 = 0\)

<table>
<thead>
<tr>
<th>Class</th>
<th>Count</th>
<th>(P(C1))</th>
<th>(P(C2))</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1</td>
<td>1/6</td>
<td>5/6</td>
<td>0.65</td>
</tr>
<tr>
<td>C2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Entropy = \(-(1/6) \log_2 (1/6) - (5/6) \log_2 (1/6) = 0.65\)

<table>
<thead>
<tr>
<th>Class</th>
<th>Count</th>
<th>(P(C1))</th>
<th>(P(C2))</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>2</td>
<td>2/6</td>
<td>4/6</td>
<td>0.92</td>
</tr>
<tr>
<td>C2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Entropy = \(-(2/6) \log_2 (2/6) - (4/6) \log_2 (4/6) = 0.92\)
Computing Information Gain After Splitting

- Information Gain:

\[
GAIN_{\text{split}} = \text{Entropy}(p) - \left( \frac{1}{n} \sum_{i=1}^{k} n_i \text{Entropy}(i) \right)
\]

Parent Node, \( p \) is split into \( k \) partitions;
\( n_i \) is number of records in partition \( i \)

- Choose the split that achieves most reduction (maximizes \( GAIN \))
- Used in the ID3 and C4.5 decision tree algorithms

Problems with Information Gain

- Info Gain tends to prefer splits that result in large number of partitions, each being small but pure

- Customer ID has highest information gain because entropy for all the children is zero
**Gain Ratio**

- Gain Ratio:

\[ \text{GainRATIO} = \frac{\text{GAIN}}{\text{SplitINFO}} \]

\[ \text{SplitINFO} = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n} \]

Parent Node, p is split into k partitions

- Adjusts Information Gain by the entropy of the partitioning (SplitINFO).
  - Higher entropy partitioning (large number of small partitions) is penalized!
- Used in C4.5 algorithm
- Designed to overcome the disadvantage of Information Gain

---

**Measure of Impurity: Classification Error**

- Classification error at a node t:

\[ \text{Error}(t) = 1 - \max_{i} P(i | t) \]

- Maximum (1 - 1/n_c) when records are equally distributed among all classes, implying least interesting information
- Minimum (0) when all records belong to one class, implying most interesting information
Computing Error of a Single Node

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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

\[ P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1 \]

Error = 1 – max (0, 1) = 1 – 1 = 0

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

\[ P(C1) = 1/6 \quad P(C2) = 5/6 \]

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

\[ P(C1) = 2/6 \quad P(C2) = 4/6 \]

Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Comparison among Impurity Measures

For a 2-class problem:

- **Entropy**
- **Gini**
- **Misclassification error**

![Graph showing entropy, Gini, and misclassification error curves]
### Misclassification Error vs Gini Index

#### Decision Tree

- **Parent:**
  - C1: 7
  - C2: 3
  - Gini = 0.42

#### Gini Index Calculation

- **Node N1 (A = Yes):**
  - Gini(N1) = 1 – (3/3)^2 – (0/3)^2
  - Gini(N1) = 0

- **Node N2 (A = No):**
  - Gini(N2) = 1 – (4/7)^2 – (3/7)^2
  - Gini(N2) = 0.489

#### Gini Index of Children

- **Gini(Children):**
  - Gini(Children) = 3/10 * 0 + 7/10 * 0.489
  - Gini(Children) = 0.342

Gini improves but error remains the same!!

---

### 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