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 x-rays or 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

Figure 3.3. General framework for building a classification model.
Classification Techniques

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

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>120K</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>Yes</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
- Marital Status: Single, Married, Divorced
- Income: < 80K, > 80K

Training Data
Model: Decision Tree
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>
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>

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>Yes</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

2/1/2021

Introduction to Data Mining, 2nd Edition
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>

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"
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 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>

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

Decision Tree Classification Task

```
\begin{tabular}{|c|c|c|c|c|}
\hline
Tid & Attrib1 & Attrib2 & Attrib3 & Class \\
\hline
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 & 65K     & No     \\
7   & Yes   & Large  & 220K    & No     \\
8   & No    & Small  & 85K     & Yes    \\
9   & No    & Medium & 75K     & No     \\
10  & No    & Small  & 90K     & Yes    \\
\hline
\end{tabular}
```

Training Set

```
\begin{tabular}{|c|c|c|c|c|}
\hline
Tid & Attrib1 & Attrib2 & Attrib3 & Class \\
\hline
11  & No      & Small   & 55K     & ?      \\
12  & Yes     & Medium  & 80K     & ?      \\
13  & Yes     & Large   & 110K    & ?      \\
14  & No      & Small   & 95K     & ?      \\
15  & No      & Large   & 67K     & ?      \\
\hline
\end{tabular}
```

Test Set

```
\begin{itemize}
  \item Induction
  \item Learn Model
  \item Decision Tree
  \item Apply Model
  \item Deduction
\end{itemize}
```
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 to 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

\[
\text{Defaulted = No} \quad \begin{array}{c}
(7,3) \\
(a)
\end{array}
\]

Hunt’s Algorithm

\[
\text{Defaulted = No} \quad \begin{array}{c}
(7,3) \\
(a)
\end{array}
\]

\[
\text{Defaulted = No} \quad \begin{array}{c}
(3,0) \\
(4,3)
\end{array}
\]

\[
\text{Defaulted = No} \quad \begin{array}{c}
(7,3) \\
(3,0) \\
(4,3)
\end{array}
\]

\[
\text{ID} \quad \text{Home Owner} \quad \text{Marital Status} \quad \text{Annual Income} \quad \text{Defaulted Borrower}
\begin{array}{cccc}
1 & \text{Yes} & \text{Single} & 125K & \text{No} \\
2 & \text{No} & \text{Married} & 100K & \text{No} \\
3 & \text{No} & \text{Single} & 70K & \text{No} \\
4 & \text{Yes} & \text{Married} & 120K & \text{No} \\
5 & \text{No} & \text{Divorced} & 95K & \text{Yes} \\
6 & \text{No} & \text{Married} & 60K & \text{No} \\
7 & \text{Yes} & \text{Divorced} & 220K & \text{No} \\
8 & \text{No} & \text{Single} & 85K & \text{Yes} \\
9 & \text{No} & \text{Married} & 75K & \text{No} \\
10 & \text{No} & \text{Single} & 90K & \text{Yes}
\end{array}
\]
Hunt’s Algorithm

(a)  Defaulded = No
    (7,3)

(b)  Defaulded = No
    (3,0)
    Defaulded = No
    (4,3)

(c)  Home Owner
    Defaulded = No
    (3,0)
    Married Status
    Defaulded = Yes
    (1,3)
    Defaulded = No
    (3,0)

2/1/2021 Introduction to Data Mining, 2nd Edition 19

Hunt’s Algorithm

(a)  Defaulded = No
    (7,3)

(b)  Defaulded = No
    (3,0)
    Defaulded = No
    (4,3)

(c)  Home Owner
    Defaulded = No
    (3,0)
    Married Status
    Defaulded = Yes
    (1,3)
    Defaulded = No
    (3,0)

(d)  Home Owner
    Defaulded = No
    (3,0)
    Single Divorced
    Annual Income
    < 80K
    Defaulded = No
    (1,0)
    >= 80K
    Defaulded = Yes
    (0,3)

ID  Home Owner  Marital Status  Annual Income  Defaulded Borrower
1  Yes  Single  125K  No
2  No  Married  100K  No
3  No  Single  70K  No
4  Yes  Married  120K  No
5  No  Divorced  95K  Yes
6  No  Married  60K  No
7  Yes  Divorced  220K  No
8  No  Single  85K  Yes
9  No  Married  75K  No
10  No  Single  50K  Yes

2/1/2021 Introduction to Data Mining, 2nd Edition 20
Design Issues of Decision Tree Induction

- How should training records be split?
  - Method for expressing 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
Test Condition for Nominal Attributes

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

Binary split:
- Divides values into two subsets

Test Condition for Ordinal Attributes

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

Binary split:
- Divides values into two subsets
- Preserve order property among attribute values
Test Condition for Continuous Attributes

(i) Binary split

(ii) Multi-way split

Splitting Based on Continuous Attributes

Different ways of handling

- **Discretization** to form an ordinal categorical attribute
  
  Ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
  
  - Static – discretize once at the beginning
  - Dynamic – repeat at each node

- **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

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>

Customer ID Gender Car Type Size Size Class
1 M Family Small C0
2 M Sports Medium C0
3 M Sports Large C0
4 M Sports Extra Large C0
5 M Sports Extra Large C0
6 M Sports Extra Large C0
7 F Sports Small C0
8 F Sports Small C0
9 F Sports Medium C0
10 F Luxury Large C1
11 M Family Large C1
12 M Family Extra Large C1
13 M Family Medium C1
14 M Luxury Extra Large C1
15 F Luxury Small C1
16 F Luxury Small C1
17 F Luxury Medium C1
18 F Luxury Medium C1
19 F Luxury Medium C1
20 F Luxury Large C1
**Measures of Node Impurity**

- **Gini Index**
  
  \[ Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2 \]
  
  Where \( p_i(t) \) is the frequency of class \( i \) at node \( t \), and \( c \) is the total number of classes.

- **Entropy**
  
  \[ Entropy = -\sum_{i=0}^{c-1} p_i(t)\log_2 p_i(t) \]

- **Misclassification error**
  
  \[ Classification\ error = 1 - \max[p_i(t)] \]

**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
   - \( M \) is the weighted impurity of child nodes
3. Choose the attribute test condition that produces the highest gain

\[ Gain = P - M \]

or equivalently, lowest impurity measure after splitting (M)
Finding the Best Split

Before Splitting:

<table>
<thead>
<tr>
<th></th>
<th>C0</th>
<th>N00</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>N01</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Node N1

A?

Yes

Node N10

C0

Yes

Node N11

C1

No

Node N2

Gain = P – M1 vs P – M2

Node N20

C0

Node N21

C1

B?

Yes

Node N3

Gain = P – M1 vs P – M2

Node N30

C0

Node N31

C1

No

Node N4

Gain = P – M1 vs P – M2

Node N40

C0

Node N41

C1

Measure of Impurity: GINI

Gini Index for a given node $t$

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

Where $p_i(t)$ is the frequency of class $i$ at node $t$, and $c$ is the total number of classes

- Maximum of $1 - 1/c$ when records are equally distributed among all classes, implying the least beneficial situation for classification
- Minimum of 0 when all records belong to one class, implying the most beneficial situation for classification
- Gini index is used in decision tree algorithms such as CART, SLIQ, SPRINT
Measure of Impurity: GINI

Gini Index for a given node t:

\[
Gini \, Index = 1 - \sum_{l=0}^{c-1} p_l(t)^2
\]

- For 2-class problem (p, 1 – p):
  - GINI = 1 – p^2 – (1 – p)^2 = 2p (1-p)

Computing Gini Index of a Single Node

\[
Gini \, Index = 1 - \sum_{l=0}^{c-1} p_l(t)^2
\]

<table>
<thead>
<tr>
<th>C1</th>
<th>0</th>
<th>C1</th>
<th>1</th>
<th>C1</th>
<th>2</th>
<th>C1</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>6</td>
<td>C2</td>
<td>5</td>
<td>C2</td>
<td>4</td>
<td>C2</td>
<td>3</td>
</tr>
</tbody>
</table>

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

Gini = 1 – P(C1)^2 – P(C2)^2 = 1 – 0 – 1 = 0

<table>
<thead>
<tr>
<th>C1</th>
<th>1</th>
<th>C1</th>
<th>1/6</th>
<th>C1</th>
<th>2/6</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>5</td>
<td>C2</td>
<td>4/6</td>
<td>C2</td>
<td>4/6</td>
</tr>
</tbody>
</table>

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

Gini = 1 – (1/6)^2 – (5/6)^2 = 0.278

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

Gini = 1 – (2/6)^2 – (4/6)^2 = 0.444
Computing Gini Index for a Collection of Nodes

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

$$\text{GINI}_{\text{split}} = \sum_{i=1}^{k} \frac{n_i}{n} \text{GINI}(i)$$

where, $n_i =$ number of records at child $i$,
$n =$ number of records at parent node $p$.

Binary Attributes: Computing GINI Index

Splits into two partitions (child nodes)
Effect of Weighing partitions:
– Larger and purer partitions are sought

<table>
<thead>
<tr>
<th>Parent</th>
<th>C1</th>
<th>C2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>N1</td>
<td>N2</td>
</tr>
<tr>
<td>B?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>Node N1</td>
<td>Node N2</td>
</tr>
</tbody>
</table>

Gini(N1) = 1 – (5/6)^2 – (1/6)^2 = 0.278
Gini(N2) = 1 – (2/6)^2 – (4/6)^2 = 0.444

Weighted Gini of N1 N2
= 6/12 * 0.278 + 6/12 * 0.444
= 0.361
Gain = 0.486 – 0.361 = 0.125
Categorical Attributes: Computing Gini Index

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

<table>
<thead>
<tr>
<th>CarType</th>
<th>Family</th>
<th>Sports</th>
<th>Luxury</th>
<th>Gini</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>0.163</td>
</tr>
<tr>
<td>C2</td>
<td>3</td>
<td>0</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

Which of these is the best?

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, $A \leq v$ 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.
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

Sorted Values

<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>
</tr>
</thead>
<tbody>
<tr>
<td>Annual Income</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>
</tr>
</tbody>
</table>

Split Positions

<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>
</tr>
</thead>
<tbody>
<tr>
<td>Annual Income</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
</tr>
<tr>
<td>55</td>
<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>
</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>
</tr>
</thead>
<tbody>
<tr>
<td>Annual Income</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>
</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>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Yes</th>
<th>0</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Gini</td>
<td>0.343</td>
<td>0.417</td>
</tr>
</tbody>
</table>

2/1/2021 Introduction to Data Mining, 2nd Edition
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

Measure of Impurity: Entropy

Entropy at a given node $t$

$$Entropy = - \sum_{i=0}^{c-1} p_i(t) \log_2 p_i(t)$$

Where $p_i(t)$ is the frequency of class $i$ at node $t$, and $c$ is the total number of classes

- Maximum of $\log_2 c$ when records are equally distributed among all classes, implying the least beneficial situation for classification
- Minimum of 0 when all records belong to one class, implying most beneficial situation for classification

-- Entropy based computations are quite similar to the GINI index computations
Computing Entropy of a Single Node

\[ Entropy = - \sum_{t=0}^{c-1} p_t(t) \log_2 p_t(t) \]

| C1 | 0   | P(C1) = 0/6 = 0   | P(C2) = 6/6 = 1   |
| C2 | 6   | Entropy = – 0 \log 0 – 1 \log 1 = – 0 – 0 = 0 |

| C1 | 1   | P(C1) = 1/6        | P(C2) = 5/6        |
| C2 | 5   | Entropy = – (1/6) \log_2 (1/6) – (5/6) \log_2 (1/6) = 0.65 |

| C1 | 2   | P(C1) = 2/6        | P(C2) = 4/6        |
| C2 | 4   | Entropy = – (2/6) \log_2 (2/6) – (4/6) \log_2 (4/6) = 0.92 |

Computing Information Gain After Splitting

Information Gain:

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

Parent Node, \( p \) is split into \( k \) partitions (children)
\( n_i \) is number of records in child node \( i \)

- Choose the split that achieves most reduction (maximizes GAIN)
- Used in the ID3 and C4.5 decision tree algorithms
- Information gain is the mutual information between the class variable and the splitting variable
Problem with large number of partitions

Node impurity measures tend 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:

\[
\text{Gain Ratio} = \frac{\text{Gain}^{\text{split}}}{\text{Split Info}} \quad \text{Split Info} = - \sum_{i=1}^{k} \frac{n_i}{n} \log_2 \frac{n_i}{n}
\]

Parent Node, \( p \) is split into \( k \) partitions (children)

- Adjusts Information Gain by the entropy of the partitioning (\( \text{Split Info} \)).
  - Higher entropy partitioning (large number of small partitions) is penalized!
- Used in C4.5 algorithm
- Designed to overcome the disadvantage of Information Gain
Gain Ratio

Gain Ratio:

\[ \text{Gain Ratio} = \frac{\text{Gain}_{\text{split}}}{\text{Split Info}} \]

\[ \text{Split Info} = \sum_{i=1}^{k} \frac{n_i}{n} \log_2 \frac{n_i}{n} \]

Parent Node, \( p \) is split into \( k \) partitions (children)

\( n_i \) is number of records in child node \( i \)

<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>8</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>3</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>Gini</td>
<td>0.163</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( \text{Split INFO} = 1.52 \)

<table>
<thead>
<tr>
<th>CarType</th>
<th>(Sports, Luxury)</th>
<th>(Family)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>Gini</td>
<td>0.468</td>
<td></td>
</tr>
</tbody>
</table>

\( \text{Split INFO} = 0.72 \)

<table>
<thead>
<tr>
<th>CarType</th>
<th>(Sports)</th>
<th>(Family, Luxury)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>C2</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>Gini</td>
<td>0.167</td>
<td></td>
</tr>
</tbody>
</table>

\( \text{Split INFO} = 0.97 \)

Measure of Impurity: Classification Error

Classification error at a node \( t \)

\[ \text{Error}(t) = 1 - \max_i [p_i(t)] \]

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

\[ Error(t) = 1 - \max_i [p_i(t)] \]

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

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

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

Comparison among Impurity Measures

For a 2-class problem:

![Graph showing entropy, Gini, and misclassification error](image.png)
Misclassification Error vs Gini Index

<table>
<thead>
<tr>
<th>Parent</th>
<th>C1</th>
<th>C2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td><strong>Gini</strong></td>
<td><strong>0.42</strong></td>
<td></td>
</tr>
</tbody>
</table>

Gini(N1) = 1 – (3/3)^2 – (0/3)^2 = 0
Gini(N2) = 1 – (4/7)^2 – (3/7)^2 = 0.489

\[
\text{Gini(N1)} = 1 - \left(\frac{3}{3}\right)^2 - \left(\frac{0}{3}\right)^2 = 0
\]
\[
\text{Gini(N2)} = 1 - \left(\frac{4}{7}\right)^2 - \left(\frac{3}{7}\right)^2 = 0.489
\]

\[
\text{Gini(Children)} = \frac{3}{10} 	imes 0 + \frac{7}{10} 	imes 0.489 = 0.342
\]

Gini improves but error remains the same!!

Misclassification error for all three cases = 0.3!
Decision Tree Based Classification

Advantages:
- Relatively inexpensive to construct
- Extremely fast at classifying unknown records
- Easy to interpret for small-sized trees
- Robust to noise (especially when methods to avoid overfitting are employed)
- Can easily handle redundant attributes
- Can easily handle irrelevant attributes (unless the attributes are interacting)

Disadvantages:
- Due to the greedy nature of splitting criterion, interacting attributes (that can distinguish between classes together but not individually) may be passed over in favor of other attributes that are less discriminating.
- Each decision boundary involves only a single attribute

Handling interactions

+ : 1000 instances

Entrophy (X) : 0.99

Entrophy (Y) : 0.99

o : 1000 instances
Handling interactions

Adding Z as a noisy attribute generated from a uniform distribution

Attribute Z will be chosen for splitting!
Limitations of single attribute-based decision boundaries

Both positive (+) and negative (o) classes generated from skewed Gaussians with centers at (8,8) and (12,12) respectively.