Classification - Basic Concepts, Decision Trees, and Model Evaluation

Lecture Notes for Chapter 4

Slides by Tan, Steinbach, Kumar adapted by Michael Hahsler

Look for accompanying R code on the course web site.
Topics

• Introduction
• Decision Trees
  – Overview
  – Tree Induction
  – Overfitting and other Practical Issues
• Model Evaluation
  – Metrics for Performance Evaluation
  – Methods to Obtain Reliable Estimates
  – Model Comparison (Relative Performance)
• Feature Selection
• Class Imbalance
Classification: Definition

- Given a collection of records (training set)
  - Each record contains a set of attributes, one of the attributes is the class.
- Find a model for class attribute as a function of the values of other attributes.
- Goal: previously unseen records should be assigned a class as accurately as possible.
  - A test set is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.
Illustrating Classification Task

<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>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>95K</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>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>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>

$y = f(X)$

**Figure 4.3.** General approach for building a classification model.
Examples of Classification Task

- Predicting tumor cells as benign or malignant

- Classifying credit card transactions as legitimate or fraudulent

- Classifying secondary structures of protein as alpha-helix, beta-sheet, or random coil

- Categorizing news stories as finance, weather, entertainment, sports, etc
Classification Techniques

• Decision Tree based Methods
• Rule-based Methods
• Memory based reasoning
• Neural Networks / Deep Learning
• Naïve Bayes and Bayesian Belief Networks
• Support Vector Machines
Topics

• Introduction
• Decision Trees
  - Overview
  - Tree Induction
  - Overfitting and other Practical Issues
• Model Evaluation
  - Metrics for Performance Evaluation
  - Methods to Obtain Reliable Estimates
  - Model Comparison (Relative Performance)
• Feature Selection
• Class Imbalance
## Example of a Decision Tree

<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>
</tbody>
</table>

### Splitting Attributes

- **Refund**: Yes → No
- **MarSt**: Married → Divorced, Single
  - TaxInc: < 80K → NO
  - TaxInc: > 80K → YES

**Model**: Decision Tree
Another Example of Decision Tree

<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>
</tbody>
</table>

There could be more than one tree that fits the same data!
Figure 4.3. General approach for building a classification model.
Apply Model to Test Data

Start from the root of tree.

Test Data

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

Refund

Yes

No

MarSt

Single, Divorced

Married

TaxInc

< 80K

> 80K

NO

NO

YES
Apply Model to Test Data

Test Data

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

Refund

MarSt

Married

< 80K

> 80K

NO

YES
Apply Model to Test Data

Test Data

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

Refund

MarSt

TaxInc
Apply Model to Test Data

Test Data

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

Refund: Yes

Marital Status: Single, Divorced

Taxable Income: < 80K
  - No
  - Yes

Taxable Income: > 80K
  - No
  - Yes
Apply Model to Test Data

Test Data

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

Refund

Marital Status

Taxable Income

Cheat

Married

< 80K

> 80K

NO

YES

Single, Divorced

< 80K

> 80K

NO

YES
Apply Model to Test Data

Refund

Yes

No

MarSt

Single, Divorced

Married

TaxInc

< 80K

> 80K

NO

YES

Test Data

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

Assign Cheat to “No”
Decision Tree: Induction

Training Set

<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>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>95K</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>

Test Set

<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>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>

Figure 4.3. General approach for building a classification model.
Topics

• Introduction
• Decision Trees
  – Overview
  – Tree Induction
  – Overfitting and other Practical Issues
• Model Evaluation
  – Metrics for Performance Evaluation
  – Methods to Obtain Reliable Estimates
  – Model Comparison (Relative Performance)
• Feature Selection
• Class Imbalance
Decision Tree Induction

- Many Algorithms:
  - Hunt’s Algorithm (one of the earliest)
  - CART (Classification And Regression Tree)
  - ID3, C4.5, C5.0 (by Ross Quinlan, information gain)
  - CHAID (CHi-squared Automatic Interaction Detection)
  - MARS (Improvement for numerical features)
  - SLIQ, SPRINT
  - Conditional Inference Trees (recursive partitioning using statistical tests)
General Structure of Hunt’s Algorithm

- "Use attributes to split the data recursively, till each split contains only a single class."

- 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 $D_t$ is an empty set, then $t$ is a leaf node labeled by the default class, $y_d$
  - If $D_t$ contains records that belong to more than one class, use an attribute test to split the data into smaller subsets. Recursively apply the procedure to each subset.
Example 2: Creating a Decision Tree
Example 2: Creating a Decision Tree

- **x1**
- **x2**
- **x**
- **o**
- **Blue circle**
- **Mixed**
- **X2 > 2.5**
- **False**
- **True**
Example 2: Creating a Decision Tree

\[
x_1 \quad x_2
\]

\[
\begin{align*}
X_2 &> 2.5 \\
\text{False} & \quad \text{True} \\
& \quad \text{Blue circle} \\
& \quad \text{Mixed}
\end{align*}
\]
Example 2: Creating a Decision Tree

Given data points with coordinates $(x_1, x_2)$, we can create a decision tree to classify them. The tree is built by splitting the data based on conditions on $x_1$ and $x_2$.

- **Condition 1**: $X_2 > 2.5$
  - **Left Branch**: False
  - **Right Branch**: True
    - **Left Leaf**: Blue circle
    - **Right Leaf**: Red X

- **Condition 2**: $X_1 > 2$
  - **Left Branch**: False
  - **Right Branch**: True
    - **Left Leaf**: Blue circle
    - **Right Leaf**: Red X

The decision tree is constructed by recursively splitting the data based on these conditions to achieve a classification.
Tree Induction

• Greedy strategy
  - Split the records based on an attribute test that optimizes a certain criterion.

• Issues
  - Determine how to split the records
    • Splitting using different attribute types?
    • How to determine the best split?
  - Determine when to stop splitting
Tree Induction

• Greedy strategy
  - Split the records based on an attribute test that optimizes a certain criterion.

• Issues
  - Determine how to split the records
    • Splitting using different attribute types?
    • How to determine the best split?
  - Determine when to stop splitting
How to Specify Test Condition?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous (interval/ratio)

- 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.

  - Family
  - Luxury
  - Sports

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

  - {Sports, Luxury} → {Family}
  - {Family, Luxury} → {Sports}

  OR

  - {Sports, Luxury} → {Family}
  - {Family, Luxury} → {Sports}
Splitting Based on Ordinal Attributes

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

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

- **What about this split?**
Splitting Based on Continuous Attributes

Binary split

Multi-way split

→ Values need to be discretized!
Splitting Based on Continuous Attributes

**Discretization** to form an ordinal categorical attribute:

- **Static** – discretize the data set once at the beginning (equal interval, equal frequency, etc.).

- **Dynamic** – discretize during the tree construction.
  
  • Example: For a binary decision: \((A < v)\) or \((A \geq v)\) consider all possible splits and finds the best cut (can be more compute intensive)
Tree Induction

• Greedy strategy
  - Split the records based on an attribute test that optimizes certain criterion.

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

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

C0: 10
C1: 10

Figure 4.12. Multiway versus binary splits.

Which test condition is the best?
How to determine the Best Split

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

- Need a measure of node impurity:

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

Non-homogeneous, High degree of impurity

Homogeneous, Low degree of impurity
Find the Best Split - General Framework

Assume we have a measure $M$ that tells us how "pure" a node is.

Before Splitting:

<table>
<thead>
<tr>
<th></th>
<th>C0</th>
<th>N00</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>N10</td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>N11</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>C1</th>
<th>N01</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>N20</td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>N21</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>C0</th>
<th>N30</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>N30</td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>N31</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>C0</th>
<th>N40</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>N40</td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>N41</td>
<td></td>
</tr>
</tbody>
</table>

Gain = $M_0 - M_{12}$ vs $M_0 - M_{34}$ → Choose best split
Measures of Node Impurity

- Gini Index
- Entropy
- Classification error
Measure of Impurity: GINI

Gini Index for a given node $t$:

$$GINI(t) = \sum_j p(j \mid t)(1 - p(j \mid t)) = 1 - \sum_j p(j \mid t)^2$$

Note: $p(j \mid t)$ is estimated as the relative frequency of class $j$ at node $t$

- Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.
- **Maximum of** $1 - 1/n_c$ (number of classes) when records are equally distributed among all classes = **maximal impurity**.
- **Minimum of** 0 when all records belong to one class = **complete purity**.
- Examples:

<table>
<thead>
<tr>
<th></th>
<th>C1</th>
<th>C2</th>
<th>Gini</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>0</td>
<td>6</td>
<td>0.000</td>
</tr>
<tr>
<td>C2</td>
<td></td>
<td></td>
<td>0.278</td>
</tr>
<tr>
<td>Gini=0.000</td>
<td>Gini=0.278</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>C1</th>
<th>C2</th>
<th>Gini</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1</td>
<td>5</td>
<td>0.444</td>
</tr>
<tr>
<td>C2</td>
<td></td>
<td></td>
<td>0.444</td>
</tr>
<tr>
<td>Gini=0.444</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>C1</th>
<th>C2</th>
<th>Gini</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>3</td>
<td>3</td>
<td>0.500</td>
</tr>
<tr>
<td>C2</td>
<td></td>
<td></td>
<td>0.500</td>
</tr>
<tr>
<td>Gini=0.500</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Examples for computing GINI

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

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

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

\[ \text{Gini} = 1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0 \]

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

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

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

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

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

\[ \text{Gini} = 1 - (2/6)^2 - (4/6)^2 = 0.444 \]

Maximal impurity here is \( 1/2 = .5 \)
Splitting Based on GINI

- When a node \( p \) is split into \( k \) partitions (children), the quality of the split is computed as a weighted sum:

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

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

- Used in CART, SLIQ, SPRINT.
Binary Attributes: Computing GINI Index

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

$$Gini(N1) = 1 - (5/8)^2 - (3/8)^2 = 0.469$$

$$Gini(N2) = 1 - (1/4)^2 - (3/4)^2 = 0.375$$

$$Gini_{Children} = \frac{8}{12} \times 0.469 + \frac{4}{12} \times 0.375 = 0.438$$

GINI improves!
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>
</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>
<tr>
<td>Gini</td>
<td>0.393</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Multi-way split

- 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>
<tr>
<td>Gini</td>
<td>0.400</td>
<td></td>
</tr>
</tbody>
</table>

<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>
<tr>
<td>Gini</td>
<td>0.419</td>
<td></td>
</tr>
</tbody>
</table>
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 $A \geq v$
- 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>
</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>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taxable Income</td>
<td></td>
<td></td>
<td></td>
<td>60</td>
<td>70</td>
<td>75</td>
<td>85</td>
<td>90</td>
<td>95</td>
<td>100</td>
<td>120</td>
</tr>
<tr>
<td>Sorted Values</td>
<td>&lt;=</td>
<td>&lt;=</td>
<td>&lt;=</td>
<td>&lt;=</td>
<td>&lt;=</td>
<td>&lt;=</td>
<td>&lt;=</td>
<td>&lt;=</td>
<td>&lt;=</td>
<td>&lt;=</td>
<td>&lt;=</td>
</tr>
<tr>
<td>&lt;=</td>
<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>
<td>230</td>
</tr>
<tr>
<td>Split Positions</td>
<td>&lt;=</td>
<td>&lt;=</td>
<td>&lt;=</td>
<td>&lt;=</td>
<td>&lt;=</td>
<td>&lt;=</td>
<td>&lt;=</td>
<td>&lt;=</td>
<td>&lt;=</td>
<td>&lt;=</td>
<td>&lt;=</td>
</tr>
<tr>
<td>Yes</td>
<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>
<tr>
<td>No</td>
<td>0</td>
<td>7</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Gini</td>
<td>0.420</td>
<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>0.420</td>
</tr>
</tbody>
</table>
Measures of Node Impurity

• Gini Index
• Entropy
• Classification error
Alternative Splitting Criteria based on INFO

Entropy at a given node $t$:

$$
\text{Entropy}(t) = - \sum_j p(j|t) \log p(j|t)
$$

NOTE: $p(j|t)$ is the relative frequency of class $j$ at node $t$

0 $\log(0) = 0$ is used!

- Measures homogeneity of a node (originally a measure of uncertainty of a random variable or information content of a message).

- **Maximum** ($\log n_c$) when records are equally distributed among all classes = **maximal impurity**.

- **Minimum** (0.0) when all records belong to one class = **maximal purity**.
Examples for computing Entropy

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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>0</td>
<td>P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1</td>
</tr>
<tr>
<td>C2</td>
<td>6</td>
<td>Entropy = − 0 \log 0 − 1 \log 1 = − 0 − 0 = 0</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1</td>
<td>P(C1) = 1/6 \quad P(C2) = 5/6</td>
</tr>
<tr>
<td>C2</td>
<td>5</td>
<td>Entropy = − (1/6) \log_2 (1/6) − (5/6) \log_2 (1/6) = 0.65</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>3</td>
<td>P(C1) = 3/6 \quad P(C2) = 3/6</td>
</tr>
<tr>
<td>C2</td>
<td>3</td>
<td>Entropy = − (3/6) \log_2 (3/6) − (3/6) \log_2 (3/6) = 1</td>
</tr>
</tbody>
</table>
Splitting Based on INFO...

Information Gain:

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

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

- Measures reduction in Entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)
- Used in ID3, C4.5 and C5.0
- **Disadvantage:** Tends to prefer splits that result in large number of partitions, each being small but pure.
Splitting Based on INFO...

Gain Ratio: 

\[
\text{GainRATIO}_{\text{split}} = \frac{GAIN_{\text{Split}}}{\text{SplitINFO}}
\]

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

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

- Adjusts Information Gain by the entropy of the partitioning (SplitINFO). **Higher entropy partitioning (large number of small partitions) is penalized!**

- Used in **C4.5**

- Designed to overcome the disadvantage of Information Gain
Measures of Node Impurity

- Gini Index
- Entropy
- Classification error
Splitting Criteria based on Classification Error

Classification error at a node $t$:

$$\text{Error}(t) = 1 - \max_i p(i|t)$$

NOTE: $p(i|t)$ is 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 = **maximal impurity** (maximal error).
- Minimum (0.0) when all records belong to one class = **maximal purity** (no error)
Examples for Computing Error

\[ \text{Error}(t) = 1 - \max_i p(i|t) \]

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

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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>3</td>
<td>P(C1) = 3/6 \quad P(C2) = 3/6</td>
</tr>
<tr>
<td>C2</td>
<td>3</td>
<td>Error = 1 - \max (3/6, 3/6) = 1 - 3/6 = .5</td>
</tr>
</tbody>
</table>
Comparison among Splitting Criteria

For a 2-class problem:
Probability of the majority class $p$ is always > .5

Note: The order is the same no matter what splitting criterion is used, however, the gain (differences) are not.
Misclassification Error vs Gini

A?

Yes

Node N1

No

Node N2

<table>
<thead>
<tr>
<th>Parent</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>7</td>
</tr>
<tr>
<td>C2</td>
<td>3</td>
</tr>
<tr>
<td>Gini</td>
<td>0.42</td>
</tr>
<tr>
<td>Error</td>
<td>0.30</td>
</tr>
</tbody>
</table>

\[
\text{Gini}(N1) = 1 - (\frac{3}{3})^2 - (\frac{0}{3})^2 = 0
\]

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

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

\[
\text{Error}(N1) = 1 - \frac{3}{3} = 0
\]

\[
\text{Error}(N2) = 1 - \frac{4}{7} = \frac{3}{7}
\]

\[
\text{Error(Split)} = \frac{3}{10} \times 0 + \frac{7}{10} \times \frac{3}{7} = 0.3
\]

Gini improves!

Error does not!!!
Tree Induction

- Greedy strategy
  - Split the records based on an attribute test that optimizes certain criterion.

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting
Stopping Criteria for Tree Induction

- Stop expanding a node when all the records belong to the same class. Happens guaranteed when there is only one observation left in the node (e.g., Hunt's algorithm).

- Stop expanding a node when all the records in the node have the same attribute values. Splitting becomes impossible.

- Early termination criterion (to be discussed later with tree pruning)
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 (improvement in Entropy).
- Handling both continuous and discrete attributes (cont. attributes are split at threshold).
- Needs entire data to fit in memory (unsuitable for large datasets).
- Trees are pruned.

Code available at
- http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz
- Open Source implementation as J48 in Weka/rWeka
Topics

• Introduction
• Decision Trees
  – Overview
  – Tree Induction
  – Overfitting and other Practical Issues
• Model Evaluation
  – Metrics for Performance Evaluation
  – Methods to Obtain Reliable Estimates
  – Model Comparison (Relative Performance)
• Feature Selection
• Class Imbalance
Underfitting and Overfitting (Example)

500 circular and 500 triangular data points.

Circular points:
\[ 0.5 \leq \sqrt{x_1^2 + x_2^2} \leq 1 \]

Triangular points:
\[ \sqrt{x_1^2 + x_2^2} > 0.5 \text{ or } \sqrt{x_1^2 + x_2^2} < 1 \]
Underfitting and Overfitting

Underfitting: when model is too simple, both training and test errors are large
Overfitting due to Noise

Decision boundary is distorted by noise point
Overfitting due to Insufficient Examples

Lack of training data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region.
Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary

- Training error does not provide a good estimate of how well the tree will perform on previously unseen records

- Need new ways for estimating errors → Generalization Error
Estimating Generalization Errors

- **Re-substitution errors**: error on training set - \( e(t) \)
- **Generalization errors**: error on testing set - \( e'(t) \)

Methods for estimating generalization errors:

- **Optimistic approach**: \( e'(t) = e(t) \)
- **Pessimistic approach**:
  - For each leaf node: \( e'(t) = (e(t) + 0.5) \)
  - Total errors: \( e'(T) = e(T) + N \times 0.5 \) (\( N \): number of leaf nodes)
  - For a tree with 30 leaf nodes and 10 errors on training (out of 1000 instances):
    Training error = \( 10/1000 = 1\% \)
    Estimated generalization error = \( (10 + 30 \times 0.5)/1000 = 2.5\% \)
- **Validation approach**:
  - uses a validation (test) data set (or cross-validation) to estimate generalization error.
Occam’s Razor (Principle of parsimony)

"Simpler is better"

• Given two models of similar generalization errors, one should prefer the simpler model over the more complex model.

• For complex models, there is a greater chance of overfitting (i.e., it fitted accidentally errors in data).

Therefore, one should include model complexity when evaluating a model
How to Address Overfitting

• Pre-Pruning (Early Stopping Rule)
  - Stop the algorithm before it becomes a fully-grown tree
  - Typical stopping conditions for a node:
    • Stop if all instances belong to the same class
    • Stop if all the attribute values are the same
  - More restrictive conditions:
    • Stop if number of instances is less than some user-specified threshold (estimates become bad for small sets of instances)
    • Stop if class distribution of instances are independent of the available features (e.g., using $\chi^2$ test)
    • Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).
How to Address Overfitting

• Post-pruning
  - Grow decision tree to its entirety
  - Try trimming sub-trees of the decision tree in a bottom-up fashion
  - If generalization error improves after trimming a sub-tree, replace the sub-tree by a leaf node (class label of leaf node is determined from majority class of instances in the sub-tree)
  - You can use MDL instead of error for post-pruning
**Refresher: Minimum Description Length (MDL)**

- \[ \text{Cost(Model,Data)} = \text{Cost(Data|Model)} + \text{Cost(Model)} \]
  - Cost is the number of bits needed for encoding.
  - Search for the least costly model.
- Cost(Data|Model) encodes the misclassification errors.
- Cost(Model) uses node encoding (number of children) plus splitting condition encoding.
Example of Post-Pruning

<p>| Class = Yes | 20 |</p>
<table>
<thead>
<tr>
<th>Class = No</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error = 10/30</td>
<td></td>
</tr>
</tbody>
</table>

Training Error (Before splitting) = 10/30

Pessimistic error = \(\frac{10 + 1 \times 0.5}{30} = \frac{10.5}{30}\)

Training Error (After splitting) = 9/30

Pessimistic error (After splitting) = \(\frac{9 + 4 \times 0.5}{30} = \frac{11}{30}\)

PRUNE!

![Decision Tree Example](image-url)
Other Issues

- Data Fragmentation
- Search Strategy
- Expressiveness
- Tree Replication
Data Fragmentation

• Number of instances gets smaller as you traverse down the tree

• Number of instances at the leaf nodes could be too small to make any statistically significant decision

→ Many algorithms stop when a node has not enough instances
Search Strategy

• Finding an optimal decision tree is NP-hard

• The algorithm presented so far uses a greedy, top-down, recursive partitioning strategy to induce a reasonable solution

• Other strategies?
  - Bottom-up
  - Bi-directional
Expressiveness

• Decision tree provides expressive representation for learning discrete-valued function
  – But they do not generalize well to certain types of Boolean functions
    • Example: parity function:
      – Class = 1 if there is an even number of Boolean attributes with truth value = True
      – Class = 0 if there is an odd number of Boolean attributes with truth value = True
    • For accurate modeling, must have a complete tree

• Not expressive enough for modeling continuous variables (cont. attributes are discretized)
Decision Boundary

- Border line between two neighboring regions of different classes is known as decision boundary.
- Decision boundary is parallel to axes because test condition involves a single attribute at-a-time.

Figure 4.20. Example of a decision tree and its decision boundaries for a two-dimensional data set.
Oblique Decision Trees

- Test condition may involve multiple attributes
- More expressive representation
- Finding optimal test condition is computationally expensive
Tree Replication

- Same subtree appears in multiple branches
- Makes the model more complicated and harder to interpret
Topics

• Introduction

• Decision Trees
  – Overview
  – Tree Induction
  – Overfitting and other Practical Issues

• Model Evaluation
  – Metrics for Performance Evaluation
  – Methods to Obtain Reliable Estimates
  – Model Comparison (Relative Performance)

• Feature Selection

• Class Imbalance
Metrics for Performance Evaluation

- Focus on the predictive capability of a model (not speed, scalability, etc.)
- Here we will focus on binary classification problems!

Confusion Matrix

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Class=Yes</td>
</tr>
<tr>
<td>Class=Yes</td>
<td>\text{a} (TP)</td>
</tr>
<tr>
<td>Class=No</td>
<td>\text{c} (FP)</td>
</tr>
</tbody>
</table>

a: TP (true positive)
b: FN (false negative)
c: FP (false positive)
d: TN (true negative)
# Metrics for Performance Evaluation

From Statistics

H0: Actual class is yes

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class=Yes</td>
<td>Class=Yes</td>
</tr>
<tr>
<td></td>
<td>Class=No</td>
</tr>
<tr>
<td>Class=Yes</td>
<td>Type I error</td>
</tr>
<tr>
<td>Class=No</td>
<td>Type II error</td>
</tr>
</tbody>
</table>

Type I error: $P(\text{NO} | \text{H0 is true}) \rightarrow \text{Significance } \alpha$

Type II error: $P(\text{Yes} | \text{H0 is false}) \rightarrow \text{Power } 1-\beta$
## Metrics for Performance Evaluation...

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Class=Yes</td>
<td>Class=No</td>
<td></td>
</tr>
<tr>
<td>Class=Yes</td>
<td>a (TP)</td>
<td>b (FN)</td>
<td></td>
</tr>
<tr>
<td>Class=No</td>
<td>c (FP)</td>
<td>d (TN)</td>
<td></td>
</tr>
</tbody>
</table>

### Most widely-used metric:

\[
\text{Accuracy} = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + TN + FP + FN}
\]

How many do we predict correct (in percent)?
Limitation of Accuracy

Consider a 2-class problem

- Number of Class 0 examples = 9990
- Number of Class 1 examples = 10

If model predicts everything to be class 0, accuracy is $9990/10000 = 99.9\%$

- Accuracy is misleading because the model does not detect any class 1 example

→ Class imbalance problem!
### Cost Matrix

Different types of error can have different cost!

| ACTUAL CLASS | PREDICTED CLASS | C(i|j) | Class=Yes | Class=No |
|--------------|-----------------|-------|-----------|----------|
| Class=Yes    | C(Yes|Yes)       | C(No|Yes) |
| Class=No     | C(Yes|No)       | C(No|No)  |

C(i|j): Cost of misclassifying class j example as class i
# Computing Cost of Classification

<table>
<thead>
<tr>
<th>Cost Matrix</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTUAL CLASS</td>
<td>C(i</td>
</tr>
<tr>
<td>+</td>
<td>-1</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
</tr>
</tbody>
</table>

**Model M_1**

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>150</td>
</tr>
<tr>
<td>-</td>
<td>60</td>
</tr>
</tbody>
</table>

Accuracy = 80%
Cost = \(-1 \times 150 + 100 \times 40 + 1 \times 60 + 0 \times 250 = 3910\)

**Model M_2**

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>250</td>
</tr>
<tr>
<td>-</td>
<td>5</td>
</tr>
</tbody>
</table>

Accuracy = 90%
Cost = 4255

Missing a + case is really bad!
## Cost vs Accuracy

<table>
<thead>
<tr>
<th>Count</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTUAL CLASS</td>
<td>Class=Yes</td>
</tr>
<tr>
<td>Class=Yes</td>
<td>a</td>
</tr>
<tr>
<td>Class=No</td>
<td>c</td>
</tr>
</tbody>
</table>

\[ N = a + b + c + d \]

\[ \text{Accuracy} = \frac{a + d}{N} \]

\[ \text{Cost} = \frac{p (a + d) + q (b + c)}{N} \]

\[ = \frac{p (a + d) + q (N - a - d)}{N} \]

\[ = q N - (q - p) (a + d) \]

\[ = N [q - (q-p) \times \text{Accuracy}] \]

Accuracy is only proportional to cost if
1. \( C(\text{Yes}|\text{No})=C(\text{No}|\text{Yes}) = q \)
2. \( C(\text{Yes}|\text{Yes})=C(\text{No}|\text{No}) = p \)
Cost-Biased Measures

Precision \( (p) = \frac{a}{a + c} \)

Recall \( (r) = \frac{a}{a + b} \)

F-measure \( (F) = \frac{2rp}{r+p} = \frac{2a}{2a+b+c} \)

- Precision is biased towards C(Yes|Yes) & C(Yes|No)
- Recall is biased towards C(Yes|Yes) & C(No|Yes)
- F-measure is biased towards all except C(No|No)

Weighted Accuracy = \( \frac{w_1 a + w_4 d}{w_1 a + w_2 b + w_3 c + w_4 d} \)
Kappa Statistic

Idea: Compare the accuracy of the classifier with a random classifier. The classifier should be better than random!

$$\kappa = \frac{\text{total accuracy} - \text{random accuracy}}{1 - \text{random accuracy}}$$

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Yes</td>
<td>Class Yes</td>
</tr>
<tr>
<td>Class Yes</td>
<td>a (TP)</td>
</tr>
<tr>
<td>Class No</td>
<td>c (FP)</td>
</tr>
</tbody>
</table>

$$\text{total accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{random accuracy} = \frac{TP + FP \cdot TN + FN + FN + TN \cdot FP + TP}{(TP + TN + FP + FN)^2}$$
ROC (Receiver Operating Characteristic)

- Developed in 1950s for signal detection theory to analyze noisy signals to characterize the trade-off between positive hits and false alarms.

- Works only for **binary classification** (two-class problems). The classes are called the **positive** and the other is the **negative** class.

- ROC curve plots TPR (true positive rate) on the y-axis against FPR (false positive rate) on the x-axis.

- Performance of each classifier represented as a point. Changing the threshold of the algorithm, sample distribution or cost matrix changes the location of the point and forms a curve.
**ROC Curve**

- Example with 1-dimensional data set containing 2 classes (positive and negative)
- Any points located at $x > t$ is classified as positive

At threshold $t$:

TPR = 0.5, FNR = 0.5, FPR = 0.12, FNR = 0.88

- Move $t$ to get the other points on the ROC curve.
(TPR, FPR):
- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (1,0): ideal

Diagonal line:
- Random guessing
- Below diagonal line:
  - prediction is opposite of the true class
Using ROC for Model Comparison

No model consistently outperform the other
- M1 is better for small FPR
- M2 is better for large FPR

Area Under the ROC curve (AUC)
- Ideal:
  - AUC = 1
- Random guess:
  - AUC = 0.5
### How to construct an ROC curve

#### Threshold at which the instance is classified -

<table>
<thead>
<tr>
<th>Class</th>
<th>+</th>
<th>-</th>
<th>+</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>+</th>
<th>-</th>
<th>+</th>
<th>+</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.25</td>
<td>0.43</td>
<td>0.53</td>
<td>0.76</td>
<td>0.85</td>
<td>0.85</td>
<td>0.85</td>
<td>0.87</td>
<td>0.93</td>
<td>0.95</td>
</tr>
<tr>
<td>TP</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>FP</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TN</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>FN</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>TPR</td>
<td>1</td>
<td>0.8</td>
<td>0.8</td>
<td>0.6</td>
<td>0.6</td>
<td>0.6</td>
<td>0.6</td>
<td>0.4</td>
<td>0.4</td>
<td>0.2</td>
</tr>
<tr>
<td>FPR</td>
<td>1</td>
<td>1</td>
<td>0.8</td>
<td>0.8</td>
<td>0.6</td>
<td>0.4</td>
<td>0.2</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

#### ROC Curve:

At a 0.23 < threshold <= 0.43
4/5 are correctly classified as +
1/5 is incorrectly classified -
Topics

- Introduction
- Decision Trees
  - Overview
  - Tree Induction
  - Overfitting and other Practical Issues
- Model Evaluation
  - Metrics for Performance Evaluation
  - Methods to Obtain Reliable Estimates
  - Model Comparison (Relative Performance)
- Class Imbalance
Learning Curve

Accuracy depends on the size of the training data.

Learning curve shows how accuracy on unseen examples changes with varying training sample size.

Variance for several runs.
Estimation Methods for the Evaluation Metric

- **Holdout**: E.g., randomly reserve 2/3 for training and 1/3 for testing.

- **Random sub-sampling**: Repeat the holdout process several times and report the average of the evaluation metric.

- **Bootstrap sampling**: Same as random subsampling, but uses sampling with replacement for the training data (sample size = n). The data not chosen for training is used for testing. Repeated several times and the average of the evaluation metric is reported.

- **Stratified sampling**: oversampling vs undersampling (to deal with class imbalance)
Estimation Methods for the Evaluation Metric

- **k-fold Cross validation** (10-fold is often used as the gold standard approach):
  - Shuffle the data
  - Partition data into k disjoint subsets
  - Repeat k times
    - Train on k-1 partitions, test on the remaining one
  - Average the results

- **Leave-one-out cross validation**: $k=n$ (used when there is not much data available)
Confidence Interval for Accuracy

• Each Prediction can be regarded as a Bernoulli trial
  - A Bernoulli trial has 2 possible outcomes:
    heads (correct) or tails (wrong)
  - Collection of Bernoulli trials has a Binomial distribution:
    • \( X \sim \text{Binomial}(N, p) \) \( X \): number of correct predictions
    • Example: Toss a fair coin 50 times, how many heads would turn up? Expected number of heads \( E[X] = N \times p = 50 \times 0.5 = 25 \)

• Given we observe \( x \) (\# of correct predictions) or equivalently, \( acc = x/N \) (\( N = \# \) of test instances):

Can we give bounds for \( p \) (true accuracy of model)?
Confidence Interval for Accuracy

- For large test sets \((N > 30)\),
  - Observed accuracy has approx. a normal distribution with mean \(p\) (true accuracy) and variance \(p(1-p)/N\)

\[
P\left(Z_{\alpha/2} < \frac{acc - p}{\sqrt{p(1-p)/N}} < Z_{1-\alpha/2}\right) = 1 - \alpha
\]

- Confidence Interval for \(p\) (the true accuracy of the model):

\[
2 \times N \times acc + Z_{\alpha/2}^2 \pm \sqrt{Z_{\alpha/2}^2 + 4 \times N \times acc - 4 \times N \times acc^2} \]
\[
= \frac{2 \times N \times acc + Z_{\alpha/2}^2 \pm \sqrt{Z_{\alpha/2}^2 + 4 \times N \times acc - 4 \times N \times acc^2}}{2 \left(N + Z_{\alpha/2}^2\right)}
\]
Confidence Interval for Accuracy

- Consider a model that produces an accuracy of 80% when evaluated on 100 test instances:
  - \( N=100, \ acc = 0.8 \)
  - Let \( 1-\alpha = 0.95 \) (95% confidence)
  - From probability table, \( Z_{\alpha/2}=1.96 \)

Using the equation from previous slide

<table>
<thead>
<tr>
<th>( N )</th>
<th>50</th>
<th>100</th>
<th>500</th>
<th>1000</th>
<th>5000</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p(\text{lower}) )</td>
<td>0.670</td>
<td>0.711</td>
<td>0.763</td>
<td>0.774</td>
<td>0.789</td>
</tr>
<tr>
<td>( p(\text{upper}) )</td>
<td>0.888</td>
<td>0.866</td>
<td>0.833</td>
<td>0.824</td>
<td>0.811</td>
</tr>
</tbody>
</table>

Table or R qnorm(1-\( \alpha/2 \))
Topics

• Introduction
• Decision Trees
  – Overview
  – Tree Induction
  – Overfitting and other Practical Issues
• Model Evaluation
  – Metrics for Performance Evaluation
  – Methods to Obtain Reliable Estimates
  – Model Comparison (Relative Performance)
• Feature Selection
• Class Imbalance
Comparing Performance of 2 Models

- Given two models, say M1 and M2, which is better?
  - M1 is tested on D1 (size=n1), found error rate = $e_1$
  - M2 is tested on D2 (size=n2), found error rate = $e_2$
  - Assume D1 and D2 are independent
  - If n1 and n2 are sufficiently large, then

$$e_1 \sim N\left(\mu_1, \sigma_1\right)$$
$$e_2 \sim N\left(\mu_2, \sigma_2\right)$$

- Approximate:

$$\hat{\sigma}_i = \frac{e_i(1-e_i)}{n_i}$$

Since they are all binominal distributions with large N
Comparing Performance of 2 Models

To test if performance difference is statistically significant: \( d = e_1 - e_2 \)

- \( d \sim N(d_t, \sigma_t) \) where \( d_t \) is the true difference
- Since D1 and D2 are independent, their variance adds up:

\[
\sigma_t^2 = \sigma_1^2 + \sigma_2^2 \approx \hat{\sigma}_1^2 + \hat{\sigma}_2^2 = \frac{e_1(1-e_1)}{n_1} + \frac{e_2(1-e_2)}{n_2}
\]

- At \((1-\alpha)\) confidence level the true difference is in the interval:

\[
d = d \pm Z_{\alpha/2} \hat{\sigma}_t
\]

- Does this interval include 0?
An Illustrative Example

- Given: M1: \( n_1 = 30, e_1 = 0.15 \)
  
  M2: \( n_2 = 5000, e_2 = 0.25 \)

- \( d = |e_2 - e_1| = 0.1 \) (2-sided test)

\[
\hat{\sigma}_d = \frac{0.15(1-0.15)}{30} + \frac{0.25(1-0.25)}{5000} = 0.0043
\]

- At 95% confidence level, \( Z_{\alpha/2} = 1.96 \)

\[
d_t = 0.100 \pm 1.96 \times \sqrt{0.0043} = 0.100 \pm 0.128
\]

\[-0.028 \leq d_t \leq 0.228\]

→ Interval contains 0 → difference is not be statistically significant!
Topics

• Introduction
• Decision Trees
  – Overview
  – Tree Induction
  – Overfitting and other Practical Issues
• Model Evaluation
  – Metrics for Performance Evaluation
  – Methods to Obtain Reliable Estimates
  – Model Comparison (Relative Performance)
• Feature Selection
• Class Imbalance
Feature Selection

What features should be used in the model?

- Univariate feature importance score
  - measures how related each feature individually is to the class variable (e.g., chi-squared statistic, information gain).

- Feature subset selection
  - tries to find the best set of features. Often uses a black box approach where different subsets are evaluated using a greedy search strategy.
Topics

- Introduction
- Decision Trees
  - Overview
  - Tree Induction
  - Overfitting and other Practical Issues
- Model Evaluation
  - Metrics for Performance Evaluation
  - Methods to Obtain Reliable Estimates
  - Model Comparison (Relative Performance)
- Feature Selection
- Class Imbalance
Class Imbalance Problem

Consider a 2-class problem
- Number of Class 0 examples = 9990
- Number of Class 1 examples = 10

If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 % and the error is only 0.1%!

Classifiers will not learn how to find examples of Class 1!
Class Imbalance Problem

Do not use accuracy for problems with strong class imbalance!

Use

- ROC curves and AUC (area under the curve)
- Precision/Recall plots or the F1 Score
- Cohen's Kappa
- Misclassification cost
Methods to Deal with Class Imbalance

- Do nothing. Sometimes you get lucky!

- **Balance the data set:** Down-sample the majority class and/or up-sample the minority class (use sampling with replacement). Synthesize new examples with SMOTE.

  This will artificially increase the error for a mistake in the minority class.
Methods to Deal with Class Imbalance

- At the algorithm level:
  - Use classifiers that predict a probability and lower the decision threshold (from .5). We can estimate probabilities for decision trees using the positive and negative training examples in each leaf node.
  - Use a cost matrix with cost-sensitive classifiers (not too many are available).
  - Use boosting techniques like AdaBoost.

- Throw away minority examples and switch to an anomaly detection framework.
Conclusion

• Classification is **supervised learning** with the goal to find a model that generalizes well.

• **Generalization error** can be estimated using test sets/cross-validation.

• Model evaluation and comparison needs to take **model complexity** into account.

• Accuracy is problematic for **imbalanced data sets**.