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 $D_t$ 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.
Hunt’s Algorithm

<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>
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
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 Specify Test Condition?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous

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

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

  ![Diagram of CarType with partitions for Family, Luxury, and Sports]

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

  ![Diagram of CarType with partitions for {Sports, Luxury} and {Family}, OR {Family, Luxury} and {Sports}]

© Tan, Steinbach, Kumar  Introduction to Data Mining  4/18/2004
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?**

![Diagram of multi-way and binary splits]

© Tan, Steinbach, Kumar
Introduction to Data Mining 4/18/2004
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
Splitting Based on Continuous Attributes

(i) Binary split

(ii) Multi-way split
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

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

- Gini Index
- Entropy
- Misclassification error
How to Find the Best Split

Before Splitting:

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

M0

A?

Yes

Node N1

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

M1

No

Node N2

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

M2

B?

Yes

Node N3

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

M3

No

Node N4

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

M4

Gain = M0 – M12 vs M0 – M34
Measure of Impurity: GINI

- Gini Index for a given node $t$:

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

(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

<table>
<thead>
<tr>
<th></th>
<th>C1</th>
<th></th>
<th>C1</th>
<th></th>
<th>C1</th>
<th></th>
<th>C1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td></td>
<td>1</td>
<td></td>
<td>2</td>
<td></td>
<td>3</td>
<td></td>
</tr>
<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>
<td></td>
</tr>
<tr>
<td>Gini=0.000</td>
<td>Gini=0.278</td>
<td>Gini=0.444</td>
<td>Gini=0.500</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Examples for computing Gini

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

<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>
Splitting Based on GINI

- Used in CART, SLIQ, SPRINT.
- When a node $p$ is split into $k$ partitions (children), the quality of split is computed as,

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

where, $n_i = \text{number of records at child } i$, $n = \text{number of records at node } p$. 
**Binary Attributes: Computing Gini Index**

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

**Gini(N1)**

\[
\text{Gini}(N1) = 1 - \left(\frac{5}{7}\right)^2 - \left(\frac{2}{7}\right)^2 = 0.408
\]

**Gini(N2)**

\[
\text{Gini}(N2) = 1 - \left(\frac{1}{5}\right)^2 - \left(\frac{4}{5}\right)^2 = 0.32
\]

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

\[
\text{Gini(Children)} = \frac{7}{12} \times 0.408 + \frac{5}{12} \times 0.32 = 0.371
\]

© Tan, Steinbach, Kumar
Categorical Attributes: Computing Gini Index

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

Multi-way split

<table>
<thead>
<tr>
<th>CarType</th>
<th>Family</th>
<th>Sports</th>
<th>Luxury</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gini</td>
<td>0.393</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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>

Taxable Income > 80K?

Yes →

No →
Continuous Attributes: Computing Gini Index...

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

<table>
<thead>
<tr>
<th>Cheat</th>
<th>No</th>
<th>No</th>
<th>No</th>
<th>Yes</th>
<th>Yes</th>
<th>Yes</th>
<th>No</th>
<th>No</th>
<th>No</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taxable Income</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>70</td>
<td>75</td>
<td>85</td>
<td>90</td>
<td>95</td>
<td>100</td>
<td>120</td>
<td>125</td>
<td>165</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>
<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>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>
</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>
</tr>
</tbody>
</table>
Alternative Splitting Criteria based on INFO

- Entropy at a given node $t$:

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

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

- Measures homogeneity of a node.
  - 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 similar to the GINI index computations
Examples for computing Entropy

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

<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>Entropy = – 0 \log 0 – 1 \log 1 = – 0 – 0 = 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>Entropy = – (1/6) \log_2 (1/6) – (5/6) \log_2 (1/6) = 0.65</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>Entropy = – (2/6) \log_2 (2/6) – (4/6) \log_2 (4/6) = 0.92</td>
</tr>
</tbody>
</table>
Splitting Based on INFO...

- Information Gain:

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

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 and C4.5
- Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.
Splitting Based on INFO...

- Gain Ratio:

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

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

Parent Node, p is split into k partitions

- 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
Splitting Criteria based on Classification Error

- Classification error at a node $t$:

$$Error(t) = 1 - \max_{i} P(i \mid t)$$

- Measures misclassification error made by a node.
  - 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
Examples for Computing Error

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

<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 Splitting Criteria

For a 2-class problem:
Misclassification Error vs Gini

Gini(N1) = 1 – (3/3)^2 – (0/3)^2 = 0

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

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

Gini improves!!
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

- Stop expanding a node when all the records have similar attribute values

- Early termination (to be discussed later)
Decision Tree Based Classification

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

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

- You can download the software from:
  [http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz](http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz)
Practical Issues of Classification

- Underfitting and Overfitting
- Missing Values
- Costs of Classification
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 data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region.

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

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

- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records

- Need new ways for estimating errors
Estimating Generalization Errors

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

**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 = $10/1000 = 1\%$
    - Generalization error = $(10 + 30 \times 0.5)/1000 = 2.5\%$
- **Reduced error pruning (REP)**:
  - uses validation data set to estimate generalization error
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
    - 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
  - Trim the nodes of the decision tree in a bottom-up fashion
  - If generalization error improves after trimming, replace sub-tree by a leaf node.
  - Class label of leaf node is determined from majority class of instances in the sub-tree
Example of Post-Pruning

Training Error (Before splitting) = 10/30
Pessimistic error = (10 + 0.5)/30 = 10.5/30
Training Error (After splitting) = 9/30
Pessimistic error (After splitting)

= (9 + 4 × 0.5)/30 = 11/30

PRUNE!
Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?

- Methods for Performance Evaluation
  - How to obtain reliable estimates?

- Methods for Model Comparison
  - How to compare the relative performance among competing models?
Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?

- Methods for Performance Evaluation
  - How to obtain reliable estimates?

- Methods for Model Comparison
  - How to compare the relative performance among competing models?
Metrics for Performance Evaluation

- Focus on the predictive capability of a model
  - Rather than how fast it takes to classify or build models, scalability, etc.

- 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>a</td>
</tr>
<tr>
<td>Class=No</td>
<td>c</td>
</tr>
</tbody>
</table>

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

<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>a (TP)</td>
</tr>
<tr>
<td>Class=No</td>
<td>c (FP)</td>
</tr>
</tbody>
</table>

- **Most widely-used metric:**

\[
\text{Accuracy} = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + TN + FP + FN}
\]
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 $\frac{9990}{10000} = 99.9\%$
  - Accuracy is misleading because model does not detect any class 1 example
Cost Matrix

| 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

#### Cost Matrix

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

#### Model M₁

<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 = 3910

#### Model M₂

<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
### Cost vs Accuracy

#### Count

<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</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>Class=No</td>
<td>c</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

Accuracy is proportional to cost if
1. \( C(Yes|No)=C(No|Yes) = q \)
2. \( C(Yes|Yes)=C(No|No) = p \)

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

Accuracy = \((a + d)/N\)

#### Cost

<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>p</td>
<td>q</td>
<td></td>
</tr>
<tr>
<td>Class=No</td>
<td>q</td>
<td>p</td>
<td></td>
</tr>
</tbody>
</table>

Cost = \( p(a + d) + q(b + c) \)

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

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

\[
= N[ q - (q-p) \times \text{Accuracy} ]
\]
Cost-Sensitive 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_1a + w_4d}{w_1a + w_2b + w_3c + w_4d}