Classification Analysis

Basic Concepts
Decision Trees
### Catching tax-evasion

<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>

Tax-return data for year 2011

<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>

A new tax return for 2012
Is this a cheating tax return?

### Classification Problem
An instance of the classification problem: learn a method for discriminating between records of different classes (cheaters vs non-cheaters)
Classification is the task of learning a target function $f$ that maps attribute set $x$ to one of the predefined class labels $y$.

One of the attributes is the class attribute in this case: Cheat.

Two class labels (or classes): Yes (1), No (0)
Why classification?

- The target function $f$ is known as a classification model.

- **Descriptive modeling**: Explanatory tool to distinguish between objects of different classes (e.g., understand why people cheat on their taxes).

- **Predictive modeling**: Predict a class of a previously unseen record.
Examples of Classification Tasks

- Predicting tumor cells as benign or malignant
- Classifying credit card transactions as legitimate or fraudulent
- Categorizing news stories as finance, weather, entertainment, sports, etc
- Identifying spam email, spam web pages, adult content
- Understanding if a web query has commercial intent or not
Training set consists of records with known class labels

Training set is used to build a classification model

A labeled test set of previously unseen data records is used to evaluate the quality of the model.

The classification model is applied to new records with unknown class labels
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>

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>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>

Test Set

Learning algorithm

Induction

Learn Model

Deduction

Apply Model
Evaluation of classification models

- Counts of **test records** that are correctly (or incorrectly) predicted by the classification model
- **Confusion matrix**

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Class = 1</th>
<th>Class = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class = 1</td>
<td>$f_{11}$</td>
<td>$f_{10}$</td>
</tr>
<tr>
<td>Class = 0</td>
<td>$f_{01}$</td>
<td>$f_{00}$</td>
</tr>
</tbody>
</table>

Accuracy = \[
\frac{\text{# correct predictions}}{\text{total # of predictions}} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}
\]

Error rate = \[
\frac{\text{# wrong predictions}}{\text{total # of predictions}} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}
\]
Classification Techniques

- Decision Tree based Methods
- Rule-based Methods
- Memory based reasoning
- Neural Networks
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines
Classification Techniques

- Decision Tree based Methods
- Rule-based Methods
- Memory based reasoning
- Neural Networks
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines
Decision Trees

- Decision tree
  - A *flow-chart-like tree* structure
  - *Internal node* denotes a *test on an attribute*
  - *Branch* represents an *outcome of the test*
  - *Leaf nodes* represent *class labels* or *class distribution*
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

Test outcome

Class labels

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

#### Data Table

<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>

#### Decision Tree

```
MarSt
  /    
/     
Married, Divorced
  |
  Yes
      NO
  No
      Refund
        /    
< 80K  > 80K
          
          TaxInc
            /    
          NO   YES
```

*There could be more than one tree that fits the same data!*
**Decision Tree Classification Task**

### 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>
Apply Model to Test Data

Start from the root of tree.

Refund

- Yes
  - NO
- No
  - MarSt
    - Single, Divorced
    - TaxInc
      - < 80K
        - NO
      - > 80K
        - YES
    - Married
      - NO

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>
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
  - NO
- No
  - MarSt
    - Single, Divorced
    - TaxInc
      - < 80K
        - NO
      - > 80K
        - YES
    - Married
      - NO
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: NO
- No:
  - MarSt:
    - Single, Divorced:
      - TaxInc:
        - < 80K: NO
        - > 80K: YES
    - Married: NO
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

< 80K

> 80K

Married

Single, Divorced

NO

YES

NO
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
  - No
    - MarSt
      - Single, Divorced
      - TaxInc
        - < 80K
          - NO
        - > 80K
          - 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>

Assign Cheat to “No”
Decision Tree 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>
Finding the best decision tree is NP-hard

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

Many Algorithms:
  - Hunt’s Algorithm (one of the earliest)
  - CART
  - ID₃, C₄.₅
  - SLIQ, SPRINT
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 with the same attribute values, then $t$ is a leaf node labeled with the majority class $y_t$.
- 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.
Constructing decision-trees (pseudocode)

\textbf{GenDecTree}(Sample } S, \text{ Features } F) \\
1. \quad \text{If } stopping\_condition(S, F) = \text{true then} \newline\quad \quad a. \quad \text{leaf} = \text{createNode()} \\
\qquad b. \quad \text{leaf.label} = \text{Classify}(S) \\
\qquad c. \quad \text{return leaf} \\
2. \quad \text{root} = \text{createNode()} \\
3. \quad \text{root.test\_condition} = \text{findBestSplit}(S, F) \\
4. \quad V = \{v | \text{v a possible outcome of root.test\_condition}\} \\
5. \quad \text{for each} \text{ value } v \in V: \newline\quad \quad a. \quad S_v = \{s | \text{root.test\_condition}(s) = v \text{ and } s \in S\}; \\
\qquad b. \quad \text{child} = \text{GenDecTree}(S_v, F); \\
\qquad c. \quad \text{Add child as a descent of root and label the edge (root} \rightarrow \text{child) as } v \\
6. \quad \text{return root}
Tree Induction

- Issues
  - How to **Classify** a leaf node
    - Assign the *majority class*
    - If leaf is empty, assign the *default class* – the class that has the highest popularity.
  - 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.

- **Binary split:** Divides values into two subsets. Need to find optimal partitioning.
Splitting Based on Ordinal Attributes

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

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

- What about this split?
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
How to determine the Best Split

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

Own Car?
- Yes
  - C0: 6
  - C1: 4
- No
  - C0: 4
  - C1: 6

Car Type?
- Family
  - C0: 1
  - C1: 3
- Sports
  - C0: 8
  - C1: 0
- Luxury
  - C0: 1
  - C1: 7

Student ID?
- c₁
  - C0: 1
  - C1: 0
- c₁₀
  - C0: 1
  - C1: 0
- c₁₁
  - C0: 0
  - C1: 1
- c₂₀
  - C0: 0
  - C1: 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*:

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

  - Non-homogeneous, High degree of impurity
  - Homogeneous, Low degree of impurity

- Ideas?
Classification error \( (t) = 1 - \max_i [p(i|t)] \)

- Used in CART, SLIQ, SPRINT.

\[ Gini(t) = 1 - \sum_{i=1}^{c} [p(i|t)]^2 \]

- Used in ID3 and C4.5

\[ Entropy(t) = -\sum_{i=1}^{c} p(i|t) \log p(i|t) \]

\[ p(i|t) \] fraction of records associated with node \( t \) belonging to class \( i \)

Measuring Node Impurity
Gain of an attribute split: compare the impurity of the parent node with the average impurity of the child nodes

$$\Delta = I( \text{parent} ) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)$$

- Maximizing the gain $\iff$ Minimizing the weighted average impurity measure of children nodes
- If $I() = \text{Entropy}()$, then $\Delta_{\text{info}}$ is called information gain
**Example**

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

- \( P(C_1) = \frac{0}{6} = 0 \) \( P(C_2) = \frac{6}{6} = 1 \)
- **Gini** = \( 1 - P(C_1)^2 - P(C_2)^2 = 1 - 0 - 1 = 0 \)
- **Entropy** = \(-0 \log 0 - 1 \log 1 = -0 - 0 = 0\)
- **Error** = \( 1 - \max(0, 1) = 1 - 1 = 0 \)

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

- \( P(C_1) = \frac{1}{6} \) \( P(C_2) = \frac{5}{6} \)
- **Gini** = \( 1 - (\frac{1}{6})^2 - (\frac{5}{6})^2 = 0.278 \)
- **Entropy** = \(-\frac{1}{6} \log_2 \frac{1}{6} - (\frac{5}{6}) \log_2 \frac{1}{6} = 0.65\)
- **Error** = \( 1 - \max(\frac{1}{6}, \frac{5}{6}) = 1 - \frac{5}{6} = \frac{1}{6} \)

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

- \( P(C_1) = \frac{2}{6} \) \( P(C_2) = \frac{4}{6} \)
- **Gini** = \( 1 - (\frac{2}{6})^2 - (\frac{4}{6})^2 = 0.444 \)
- **Entropy** = \(-\frac{2}{6} \log_2 \frac{2}{6} - (\frac{4}{6}) \log_2 \frac{4}{6} = 0.92\)
- **Error** = \( 1 - \max(\frac{2}{6}, \frac{4}{6}) = 1 - \frac{4}{6} = \frac{1}{3} \)
Impurity measures

- All of the impurity measures take value zero \(\text{(minimum)}\) for the case of a pure node where a single value has probability 1.
- All of the impurity measures take \(\text{maximum}\) value when the class distribution in a node is \text{uniform}. 
Comparison among Splitting Criteria

For a 2-class problem:

The different impurity measures are consistent
## Categorical Attributes

- For **binary** values split in two
- For **multivalued** attributes, 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><strong>Gini</strong></td>
<td><strong>0.393</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Two-way split

- (find best partition of values)

#### CarType

<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><strong>Gini</strong></td>
<td><strong>0.400</strong></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><strong>Gini</strong></td>
<td><strong>0.419</strong></td>
<td></td>
</tr>
</tbody>
</table>
Continuous Attributes

- Use Binary Decisions based on one value
- 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 $A \geq v$
- Exhaustive method to choose best $v$
  - For each $v$, scan the database to gather count matrix and compute the impurity 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

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

<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></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>172</td>
<td>220</td>
<td></td>
</tr>
<tr>
<td>&lt;=</td>
<td>&gt;</td>
<td>&lt;=</td>
<td>&gt;</td>
<td>&lt;=</td>
<td>&gt;</td>
<td>&lt;=</td>
<td>&gt;</td>
<td>&lt;=</td>
<td>&gt;</td>
<td>&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>125</td>
<td>230</td>
<td></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>
<td>0</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><strong>0.300</strong></td>
<td>0.343</td>
<td>0.375</td>
<td>0.400</td>
<td>0.420</td>
</tr>
</tbody>
</table>
Splitting based on impurity

- Impurity measures favor attributes with large number of values

- A test condition with large number of outcomes may not be desirable
  - # of records in each partition is too small to make predictions
Splitting based on INFO

Figure 4.12. Multiway versus binary splits.
Gain Ratio

Splitting using information gain

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

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

Parent Node, p is split into k partitions
\(n_i\) 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 impurity
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)
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
Other Issues

- Data Fragmentation
- Expressiveness
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
- You can introduce a lower bound on the number of items per leaf node in the stopping criterion.
A classifier defines a function that discriminates between two (or more) classes.

The expressiveness of a classifier is the class of functions that it can model, and the kind of data that it can separate.

- When we have discrete (or binary) values, we are interested in the class of boolean functions that can be modeled.
- If the data-points are real vectors we talk about the decision boundary that the classifier can model.
• 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
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

- Less expressive for modeling continuous variables
  - Particularly when test condition involves only a single attribute at-a-time
Oblique Decision Trees

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

$\text{Class} = +$  $\text{x + y < 1}$  $\text{Class} = \bullet$
Practical Issues of Classification

- Underfitting and Overfitting
- Evaluation
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**: when model is too complex it models the details of the training set and fails on the test set
Overfitting due to Noise

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

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

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

- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records
  - The model does not generalize well

- Need new ways for estimating errors
Estimating Generalization Errors

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

**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)
    - Penalize large trees
  - 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\times0.5)/1000 = 2.5%

**Using validation set:**
- Split data into training, validation, test
- Use validation dataset to estimate generalization error
- Drawback: less data for training.
Occam’s Razor

- 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 that it was fitted accidentally by errors in data.

- Therefore, one should include model complexity when evaluating a model.
Minimum Description Length (MDL)

- \( \text{Cost(Model, Data)} = \text{Cost(Data|Model)} + \text{Cost(Model)} \)
  - Search for the least costly model.

- \( \text{Cost(Data|Model)} \) encodes the misclassification errors.
- \( \text{Cost(Model)} \) encodes the decision tree
  - node encoding (number of children) plus splitting condition encoding.
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).
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

- Can use MDL for post-pruning
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 \times 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?
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>
<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:**
  \[
  \text{Accuracy} = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + TN + FP + FN}
  \]
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

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
<th>Class=Yes</th>
<th>Class=No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class=Yes</td>
<td>C(Yes</td>
<td>Yes)</td>
<td>C(No</td>
</tr>
<tr>
<td>Class=No</td>
<td>C(Yes</td>
<td>No)</td>
<td>C(No</td>
</tr>
</tbody>
</table>

\[ C(i|j): \text{Cost of classifying class } j \text{ example as class } i \]

\[
\text{Weighted Accuracy} = \frac{w_1 a + w_4 d}{w_1 a + w_2 b + w_3 c + w_4 d}
\]
Computing Cost of Classification

Cost Matrix

| ACTUAL CLASS | PREDICTED CLASS | C(i|j) |
|--------------|-----------------|-------|
| +            | +               | -100  |
| -            | -               | 0     |

Model $M_1$

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

Accuracy = 80%
Cost = 3910

Model $M_2$

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

Accuracy = 90%
Cost = 4255
### Cost vs Accuracy

#### Count

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
<th>Class=Yes</th>
<th>Class=No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class=Yes</td>
<td></td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>Class=No</td>
<td></td>
<td>c</td>
<td>d</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>Class=Yes</th>
<th>Class=No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class=Yes</td>
<td></td>
<td>p</td>
<td>q</td>
</tr>
<tr>
<td>Class=No</td>
<td></td>
<td>q</td>
<td>p</td>
</tr>
</tbody>
</table>

- **Cost** = $p (a + d) + q (b + c)$
  - = $p (a + d) + q (N - a - d)$
  - = $q N - (q - p)(a + d)$
  - = $N [q - (q-p) \times \text{Accuracy}]$
Precision-Recall

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

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

F-measure (F) = \frac{1}{\frac{1}{r} + \frac{1}{p}} = \frac{2rp}{r + p} = \frac{2a}{2a + b + c} = \frac{2TP}{2TP + FP + FN}

- 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 \( C(No|No) \)
Usually for parameterized models, it controls the precision/recall tradeoff
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?
How to obtain a reliable estimate of performance?

Performance of a model may depend on other factors besides the learning algorithm:
- Class distribution
- Cost of misclassification
- Size of training and test sets
Methods of Estimation

- **Holdout**
  - Reserve $2/3$ for training and $1/3$ for testing
- **Random subsampling**
  - One sample may be biased -- Repeated holdout
- **Cross validation**
  - Partition data into $k$ disjoint subsets
  - $k$-fold: train on $k-1$ partitions, test on the remaining one
  - Leave-one-out: $k=n$
  - Guarantees that each record is used the same number of times for training and testing
- **Bootstrap**
  - Sampling with replacement
  - ~63% of records used for training, ~27% for testing
Dealing with class Imbalance

- If the class we are interested in is very rare, then the classifier will ignore it.
  - The class imbalance problem
- Solution
  - We can modify the optimization criterion by using a cost sensitive metric
  - We can **balance** the class distribution
    - Sample from the larger class so that the size of the two classes is the same
    - Replicate the data of the class of interest so that the classes are balanced
      - Over-fitting issues
Learning curve shows how accuracy changes with varying sample size.

Requires a sampling schedule for creating learning curve.

Effect of small sample size:
- Bias in the estimate
- Variance of estimate
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?
ROC (Receiver Operating Characteristic)

- Developed in 1950s for signal detection theory to analyze noisy signals
  - Characterize the trade-off between positive hits and false alarms
- **ROC** curve plots **TPR** (on the y-axis) against **FPR** (on the x-axis)

\[ TPR = \frac{TP}{TP + FN} \]

Fraction of positive instances predicted **correctly**

\[ FPR = \frac{FP}{FP + TN} \]

Fraction of negative instances predicted **incorrectly**

<table>
<thead>
<tr>
<th>Actual</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td>a (TP)</td>
</tr>
<tr>
<td>No</td>
<td>c (FP)</td>
</tr>
</tbody>
</table>
ROC (Receiver Operating Characteristic)

- Performance of a classifier represented as a point on the ROC curve

- Changing some parameter of the algorithm, sample distribution or cost matrix changes the location of the point
At threshold $t$:

$TP = 0.5, \ FN = 0.5, \ FP = 0.12, \ FN = 0.88$
(TP,FP):
- (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

<table>
<thead>
<tr>
<th>Actual</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>No</td>
<td>c</td>
</tr>
</tbody>
</table>

- TP (true positive)
- FP (false positive)
- TN (true negative)
- FN (false negative)
Using ROC for Model Comparison

- No model consistently outperform the other
  - \( M_1 \) is better for small FPR
  - \( M_2 \) is better for large FPR

- Area Under the ROC curve (AUC)
  - Ideal: Area = 1
  - Random guess:
    - Area = 0.5
Area Under the Curve (AUC) as a single number for evaluation
set.seed(1234)
ind = sample(2, nrow(iris), replace=TRUE, prob=c(0.7, 0.3))
trainData = iris[ind==1,]
testData = iris[ind==2,]

install.packages(c("party"))
library(party)
myFormula <- Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width
iris_ctree <- ctree(myFormula, data=trainData)

# check the prediction
table(predict(iris_ctree), trainData$Species)

print(iris_ctree)
plot(iris_ctree)
plot(iris_ctree, type="simple")