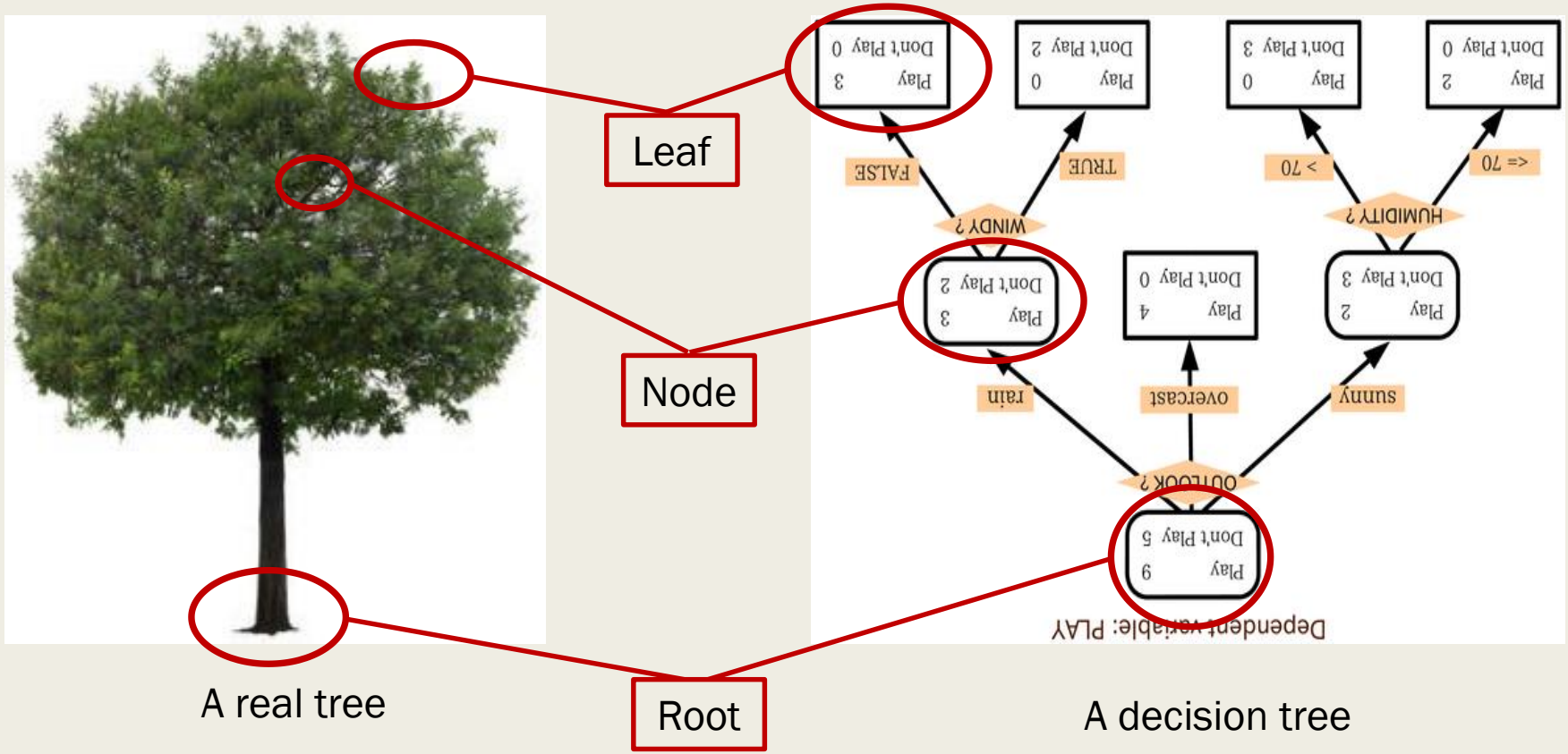
TOPIC 5. DECISION TREE

Introduction

- What are root, node, and leaf ?



Introduction

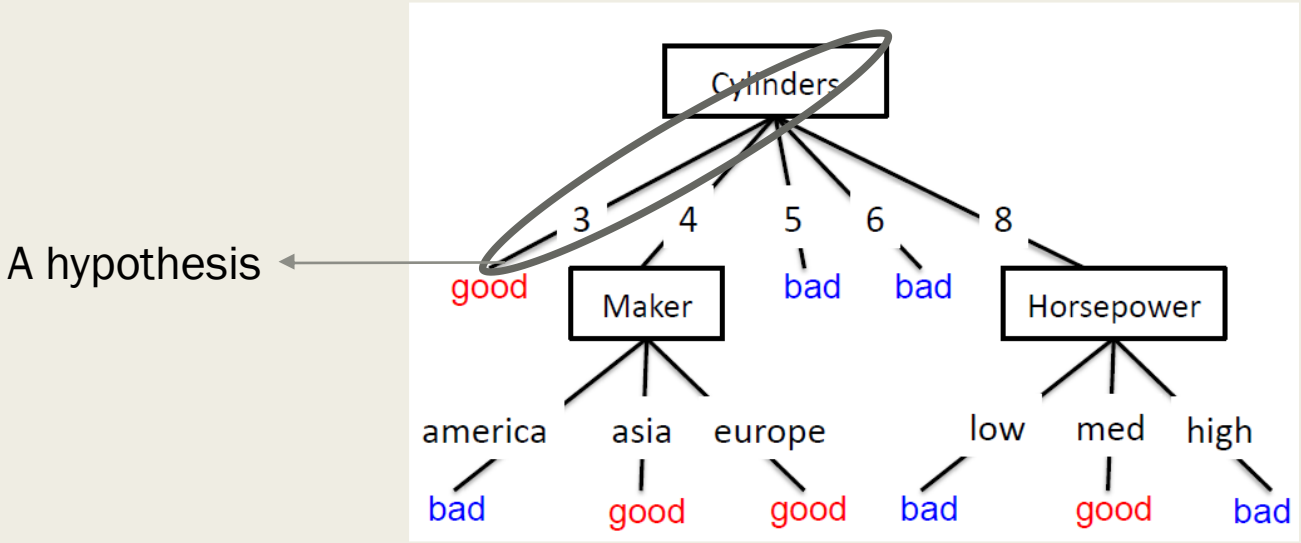
- A model defined as a function of determining output variable Y based on input variables X.
- Decision tree used for a classification problem is called classification tree, where the interested output variable Y is categorical.
- Examples of classification:

  Predicting tumor cells as benign or malignant

  Classifying credit card transactions as legitimate or fraudulent
- Decision tree can be used for predicting continuous variables, such as income or revenue, etc., which is called regression tree.
- A decision tree composed of a root, many nodes and many leafs (end nodes).

Introduction

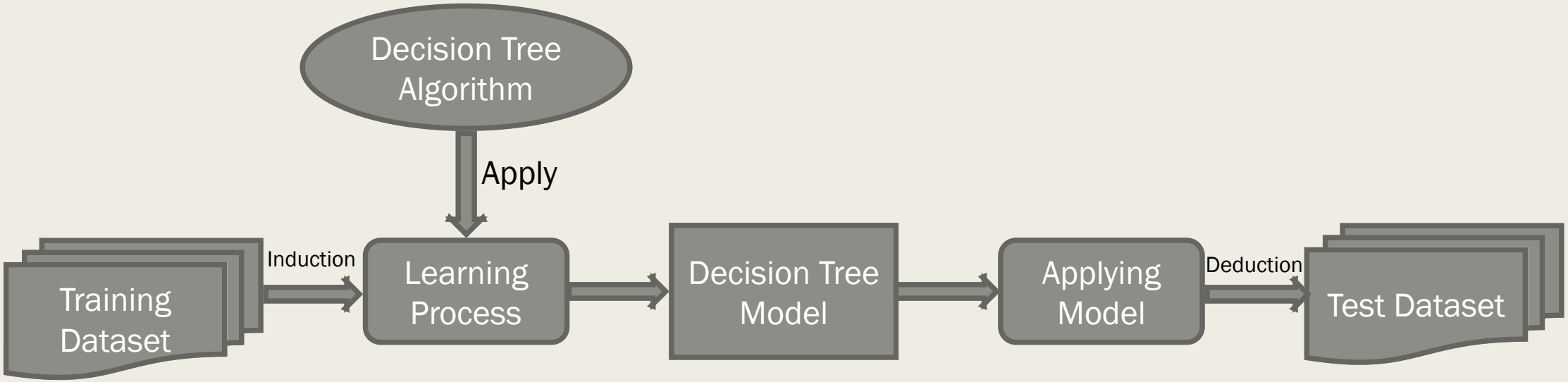
- A decision tree can be regarded as a finite sets of hypotheses organized in a tree structure for decision-making
- Each internal node tests an attribute x_i
- One branch for each possible value $x_i = v$
- Each leaf assigns a label (class) to y
- To classify input x: traverse the tree from root to leaf and output the label of y



This slide is adapted from Luke Zettlemoyer, Carlos Guestrin, and Andrew Moore

Introduction

- A Classification Process via Decision Tree



Decision 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*

Decision Tree Induction

Specify test condition

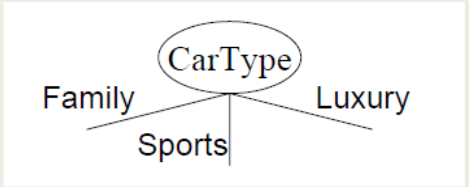
- Depends on attribute types
  - *Categorical*
  - *Continuous*

- Depends on number of ways to split
  - *2-way split*
  - *Multi-way split*

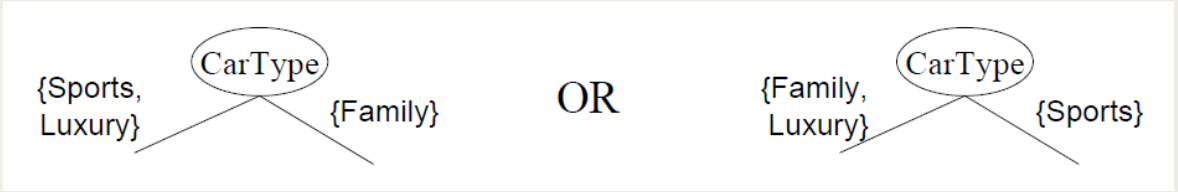
Decision Tree Induction

Split based on categorical attributes

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



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



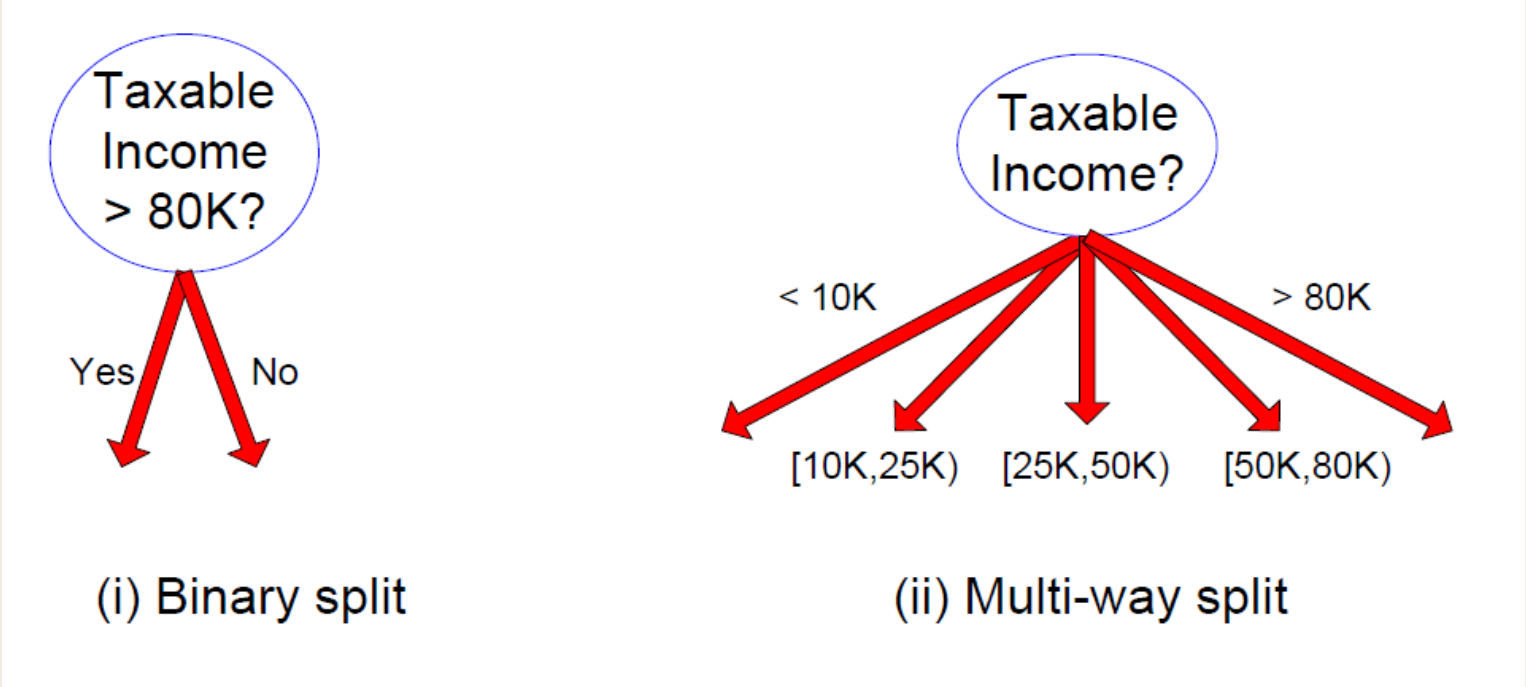
Decision Tree Induction

Split based on continuous attributes

- Multiple options of tackling such challenge
  - *Discretization to form a 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: $(X < v)$ or $(X \geq v)$*
    - Consider all possible splits and finds the best cut
    - Can be more computational intensive

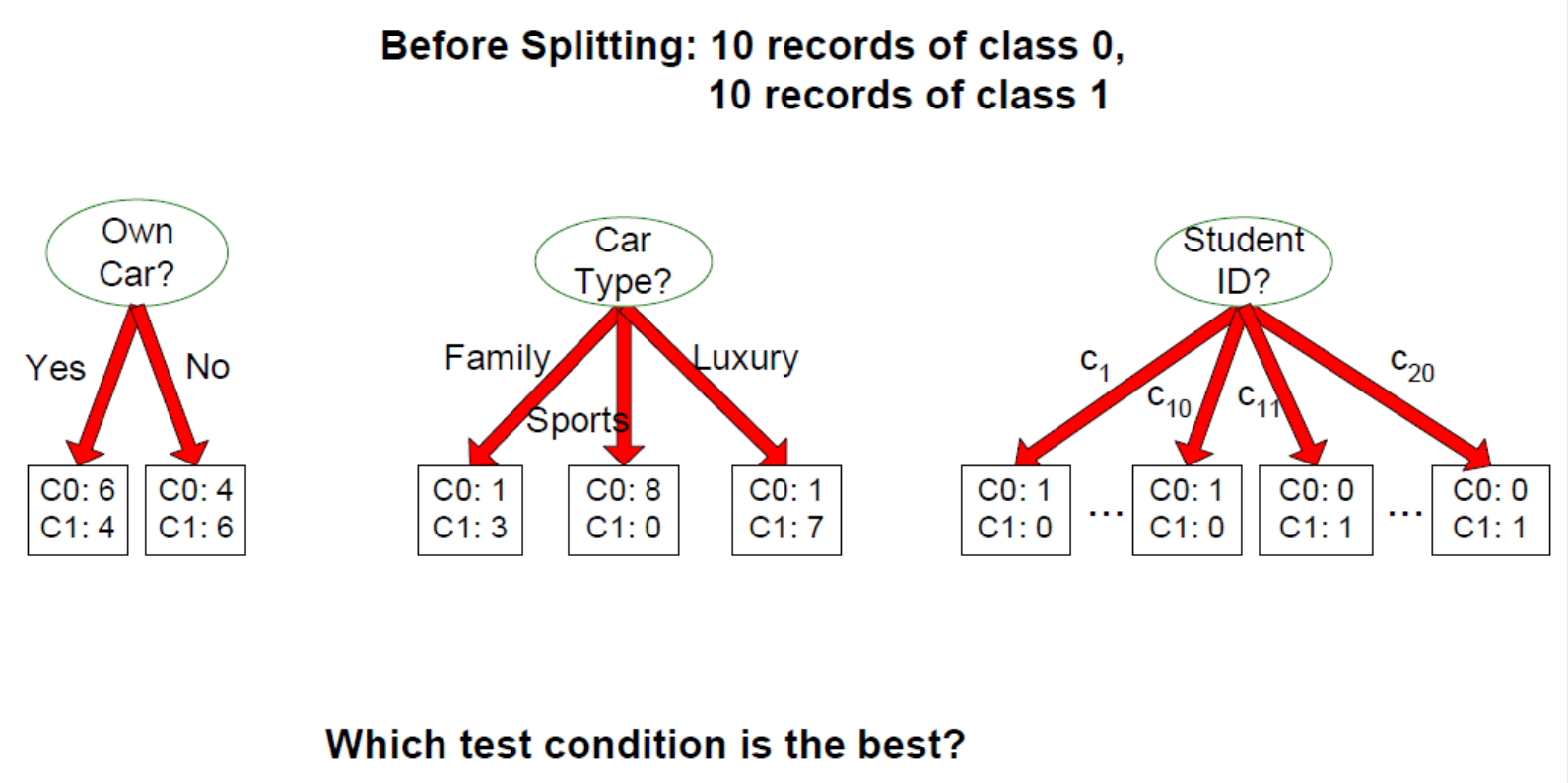
Decision Tree Induction

Splitting based on continuous attributes



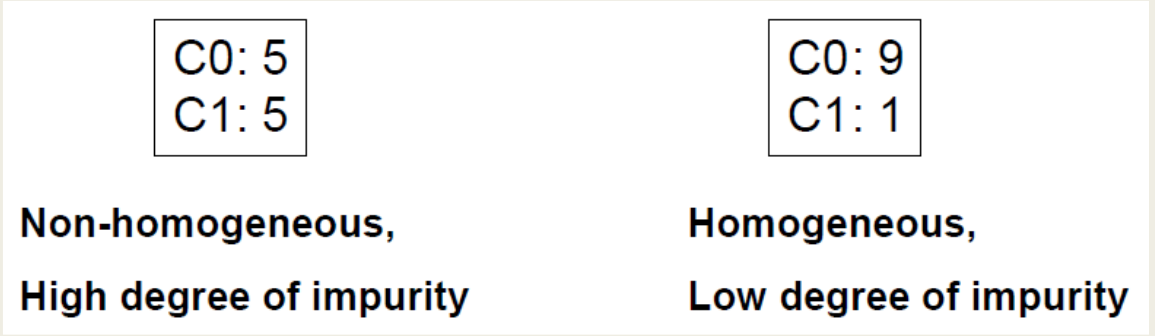
Decision Tree Induction

- How to determine the best split



Decision Tree Introduction

- Greedy approach:
  - *Nodes with homogeneous class distribution are preferred*
- Need a measure of node impurity:



Decision Tree Induction

- Maths in Decision Tree: Three measures of node impurity
  - *Gini Index*
  - *Entropy and Information Gain*
  - *Misclassification Error*

Gini Index

- Gini Index for a given node *t*:

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

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

- *Maximum (1 – 1/n_c), n_c is the number of classes, when records are equally distributed among all classes, implying least interesting information*
- *Minimum 0 when all records belong to one class, implying most interesting information*

C1	0
C2	6
Gini=0.000	

C1	1
C2	5
Gini=0.278	

C1	2
C2	4
Gini=0.444	

C1	3
C2	3
Gini=0.500	

Gini Index

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

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

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Gini = 1 – $(1/6)^2$ – $(5/6)^2$ = 0.278

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Gini = 1 – $(2/6)^2$ – $(4/6)^2$ = 0.444

Gini Index

- Multi-way split

  - 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 = number of records at child i,

n = number of records at node p.

$$G_{split} = \text{GINI(p)} - \left(\sum_{i=1}^{k} \frac{n_i}{n} GINI(i)\right)$$

Entropy and Information Gain

- Entropy at a given node t:

$$Entropy(t) = -\sum_{j} p(j \mid t) \log p(j \mid t)$$

(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 when all records belong to one class, implying most information
  - *Entropy based computations are similar to the Gini index computations*

Entropy and Information Gain

Examples of Entropy calculation

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Entropy and Information Gain

- Information Gain Definition:

$$G_{split} = Entropy(p) - (\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i))$$

Describe split the parent node p into k children; n_i is the number of records in children I

- *Measures Reduction in Entropy achieved because of the split. Choose the split that achieves most reduction, maximizing G*
- *Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure, model be too specific rather than general*

Entropy and Information Gain

- Gain Ratio:

$$GR_{split} = \frac{G_{split}}{I_{split}}, I_{split} = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

Describe split parent node p into k children, n_i is the number of records in children i

- *Adjust Information Gain by the entropy of the partitioning (I_{split}). Higher entropy partitioning (a large number of partitions with small sizes) is penalized.*
- *Designed to overcome the disadvantage of Information Gain*

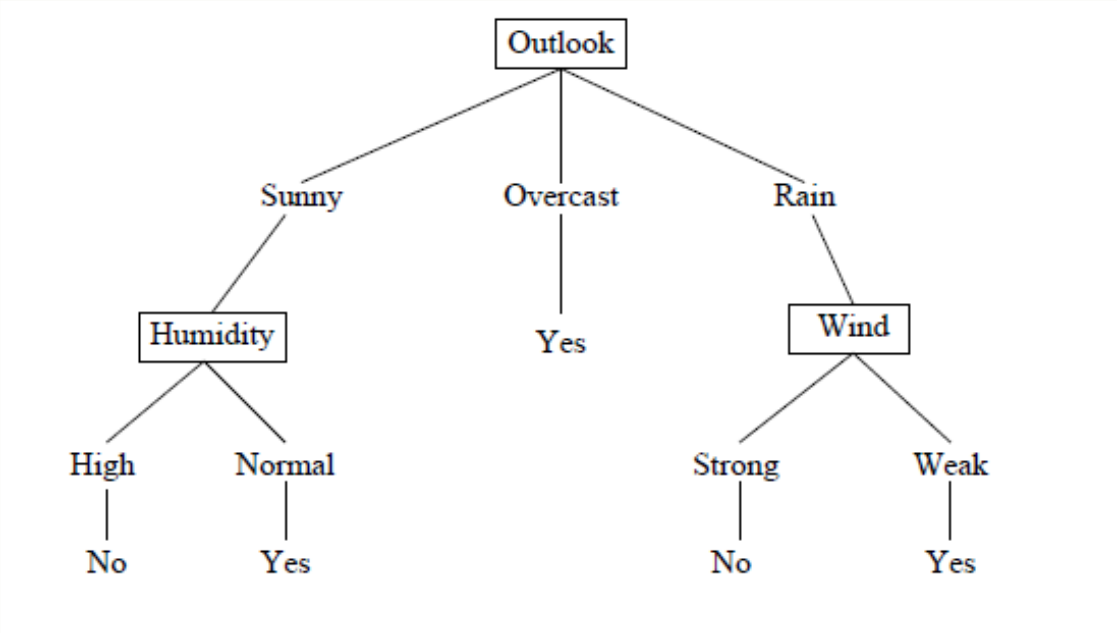
Entropy and Information Gain

Training Examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Entropy and Information Gain

Hand calculation will be provided and final result is:



Misclassification Error

- Classification error at a node t

$$Error(t) = 1 - \max_j p(j \mid t)$$

- 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 when all records belong to one class, implying most interesting information*

Misclassification Error

- Example

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Error = 1 – max (0, 1) = 1 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

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

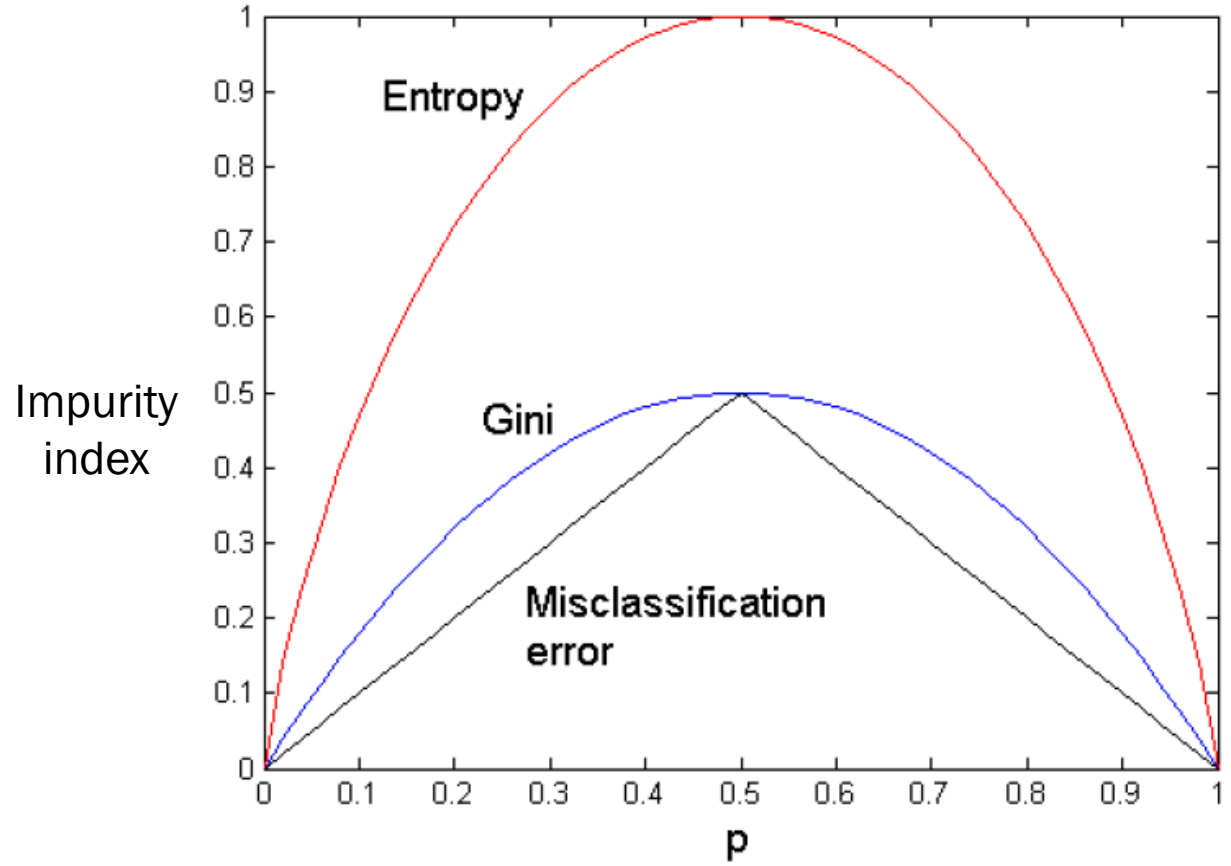
C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

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

Comparison of three measures

For a 2-class problem:



Entropy is most sensitive to the impurity while misclassification error is least sensitive in the classification, Gini stays in the middle.

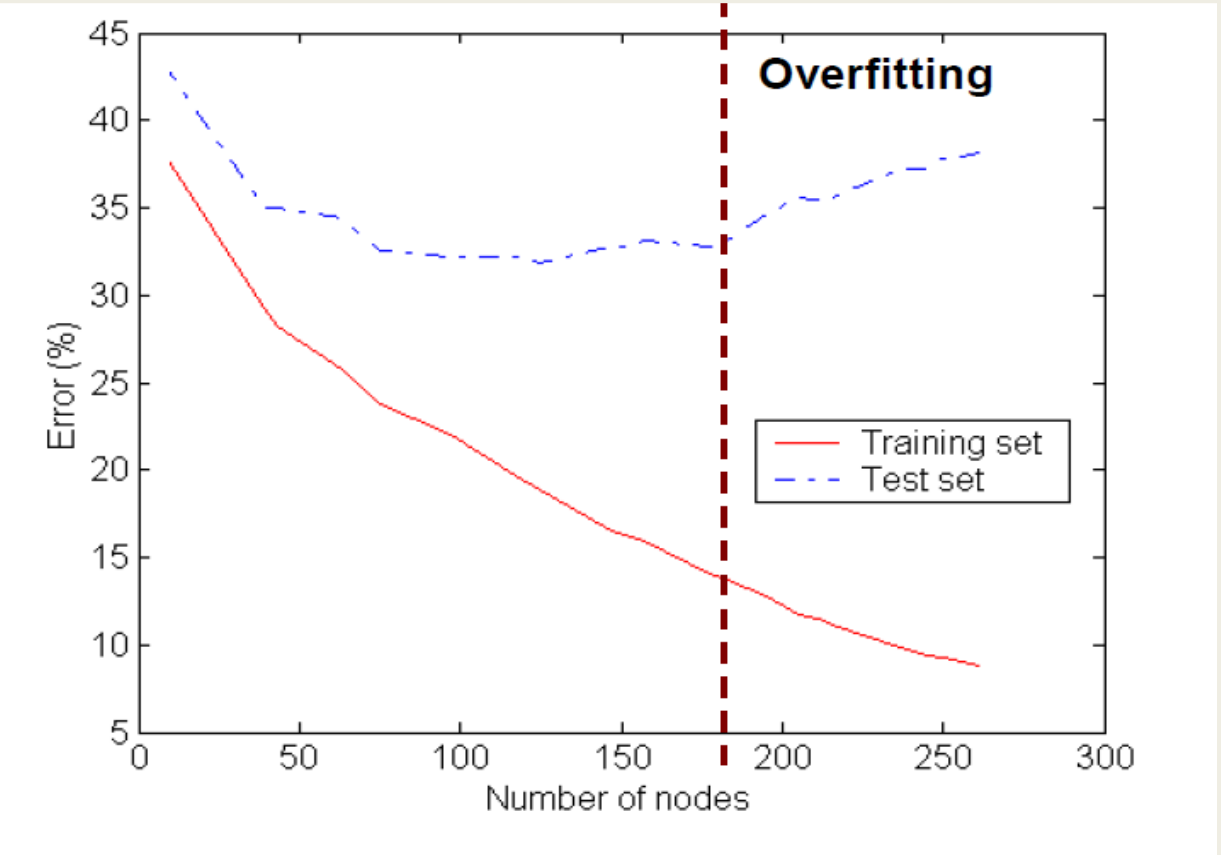
Tree Induction: Stopping Criteria

- Stop expanding a node when all records belong to the same class
- Stop expanding a node when all records have similar attribute values
- Early termination

Advantages of Decision Tree Based Classification

- 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 datasets

Underfitting and Overfitting



Underfitting: model is too simple so that both training and test errors are large
Overfitting: model is too complex so that the training error is small, but the test error is large

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
    - Stop if expanding the current node does not improve impurity measures

How to Address Overfitting

- Post-pruning
  - *Grow decision tree to its entirety*
  - *Trim the nodes of the decision tree in a bottom-up fashion*
  - *Find a number of subtrees and choose the one that has the lowest test error*
  - *Class label of leaf node is determined from majority class of instances in the sub-tree*