Classification using Decision Trees

Assoc. Prof. Kitsana Waiyamai, Ph.D.
Dept. Of Computer Engineering
Faculty of Enginerring, Kasetsart University
Bangkok, Thailand

Why Decision Tree

- Decision trees (DTs), or classification trees, are one of the most popular analytics tools (along with linear regression).
- They are:
  - Easy to understand
  - Easy to implement
  - Easy to use
  - Computationally cheap
  - Support both Classification and Prediction Task
- Almost all the data analytics tools include DTs.
- They have advantages for model comprehensibility.

Explainability of the Classification Rules

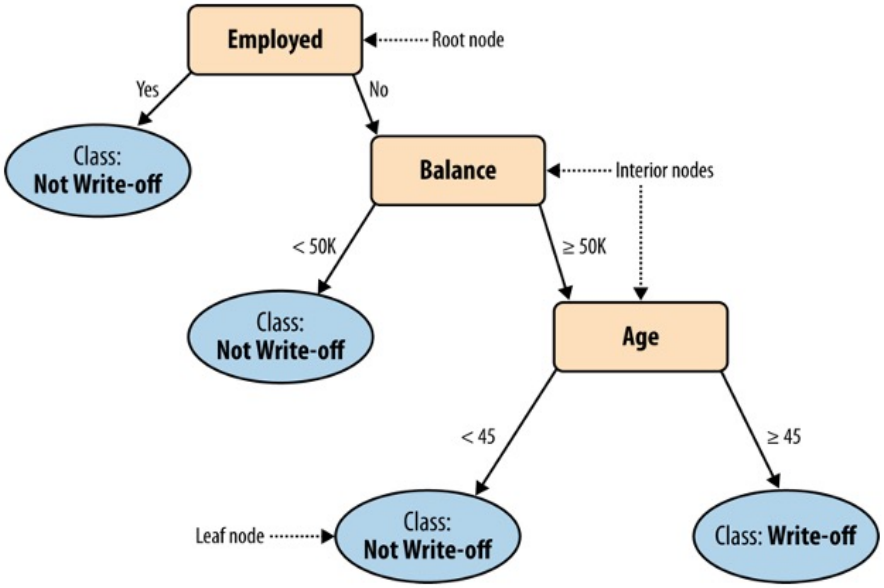
Goal: Classify or predict class of unseen data based on a set of predictors.

- The output is a set of **Classification Rules**.

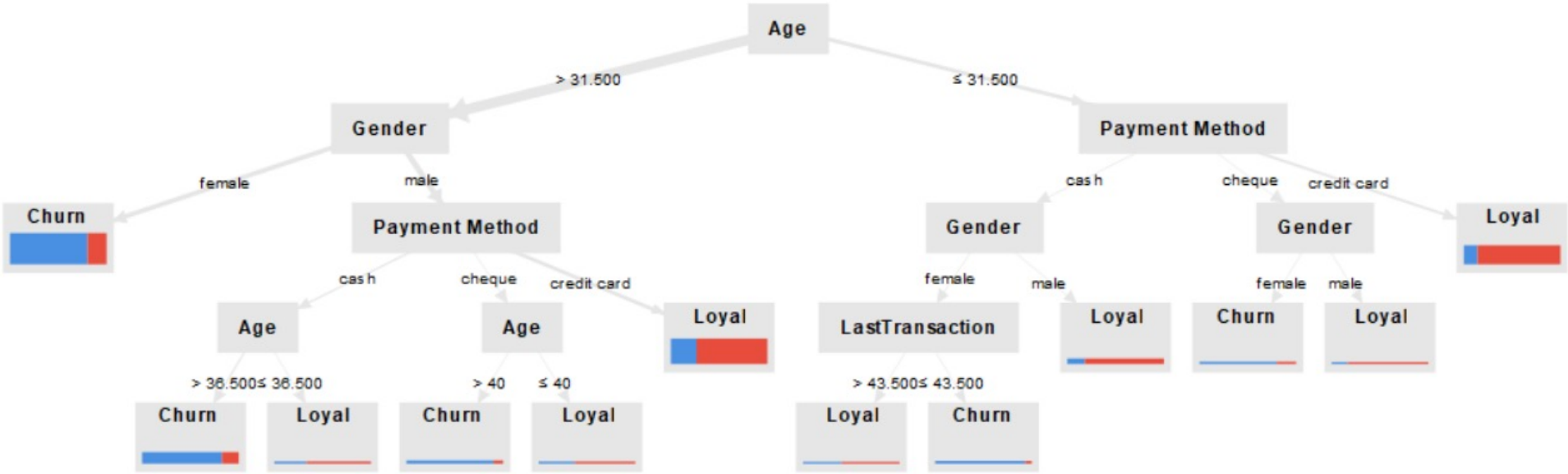
Example:

- Goal: classify a record as "will accept credit card offer" or "will not accept"
- "IF (Income > 92.5) AND (Education < 1.5) AND (Family <= 2.5) THEN Class = 0 (non-acceptor).
- Rules are represented by tree diagrams.

Example of a Decision Trees



Example of a Decision Trees



General Structure of Tree Construction

- Let D_t be the set of training records that reach a node t
- General Procedure:

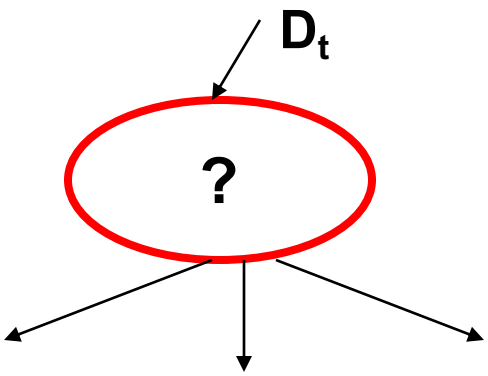
  1) Check Stopping Condition. If D_t contains records that belong the same class y_t, then t becomes a leaf node labeled as y_t and Stop tree construction. If D_t contains records that belong to more than one class, go to step 2)

  2) Test all possible conditions based on remaining attributes. Then select the one that produces the Highest Gain of Impurity Reduction.

  3) Based on the selected condition, Split the data into smaller subsets.

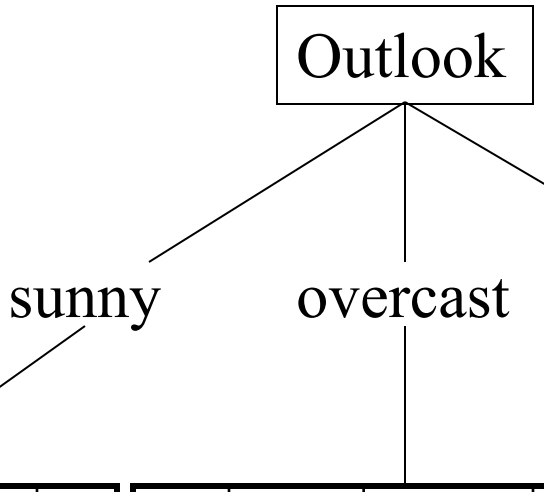
  4) Go to step 1). Each subset becomes a new current.

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Example – Branch on outlook

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N



1	Hot	High	False	N
2	Hot	High	True	N
8	Mild	High	False	N
9	Cool	Normal	False	P
11	Mild	Normal	True	P

3	Hot	High	False	P
7	Cool	Normal	true	P
12	Mild	High	True	P
13	Hot	Normal	False	P

4	Mild	High	False	P
5	Cool	Normal	False	P
6	Cool	Normal	True	N
10	Mild	Normal	False	P
14	Mild	High	True	N

Recursive Partitioning

- At each successive stage, compare all possible splits in all variables.
- Choose the split that **reduces impurity the most**
- Chosen split points become **nodes on the tree**.

Decision Tree Construction

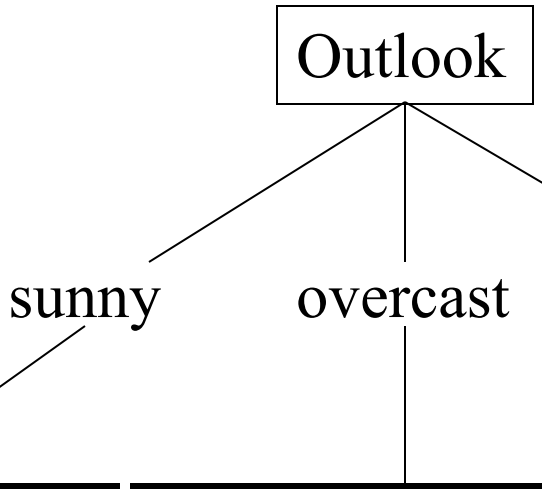
Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Atr=?

Gain(Outlook) = 0.246
Gain(Temperature) = 0.029
Gain(Humidity) = 0.151
Gain(Windy) = 0.048

Example – Branch on outlook

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

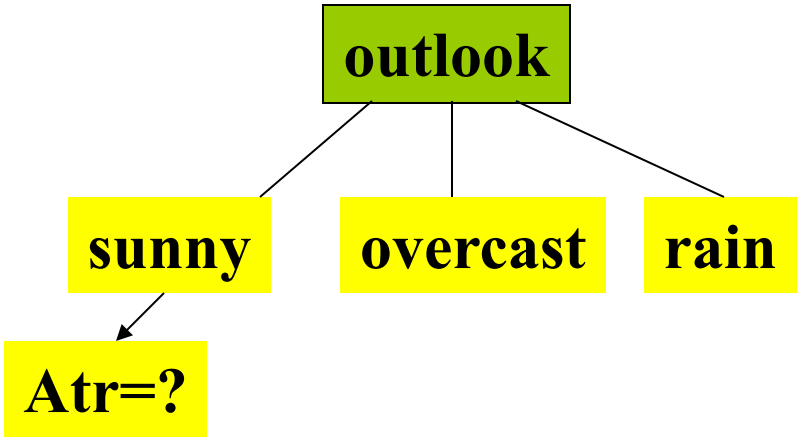


1	Hot	High	False	N
2	Hot	High	True	N
8	Mild	High	False	N
9	Cool	Normal	False	P
11	Mild	Normal	True	P

3	Hot	High	False	P
7	Cool	Normal	true	P
12	Mild	High	True	P
13	Hot	Normal	False	P

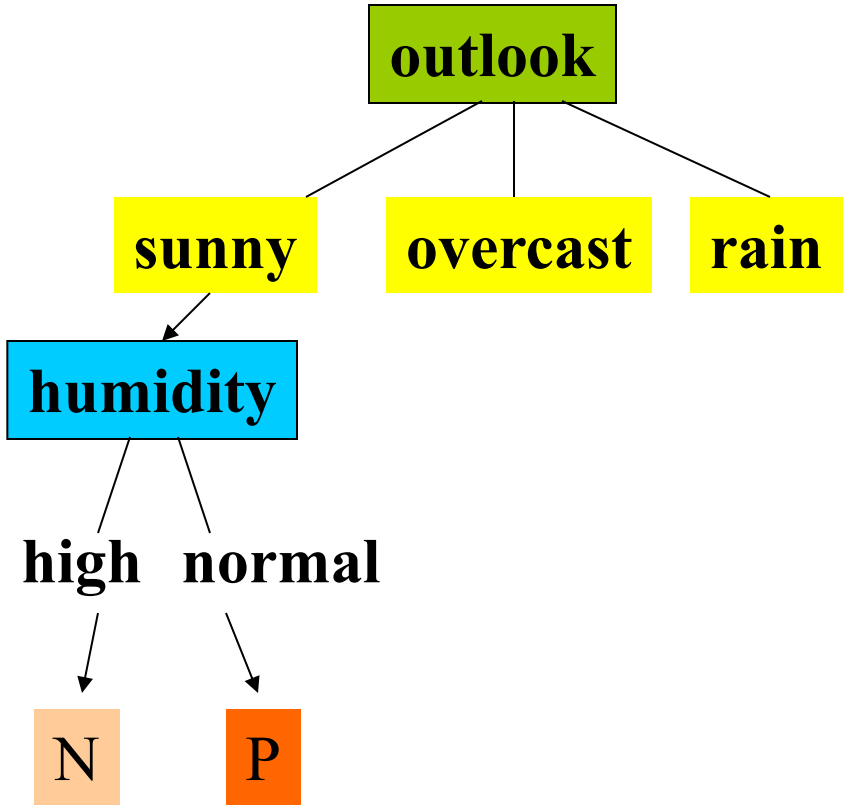
4	Mild	High	False	P
5	Cool	Normal	False	P
6	Cool	Normal	True	N
10	Mild	Normal	False	P
14	Mild	High	True	N

Decision Tree Construction



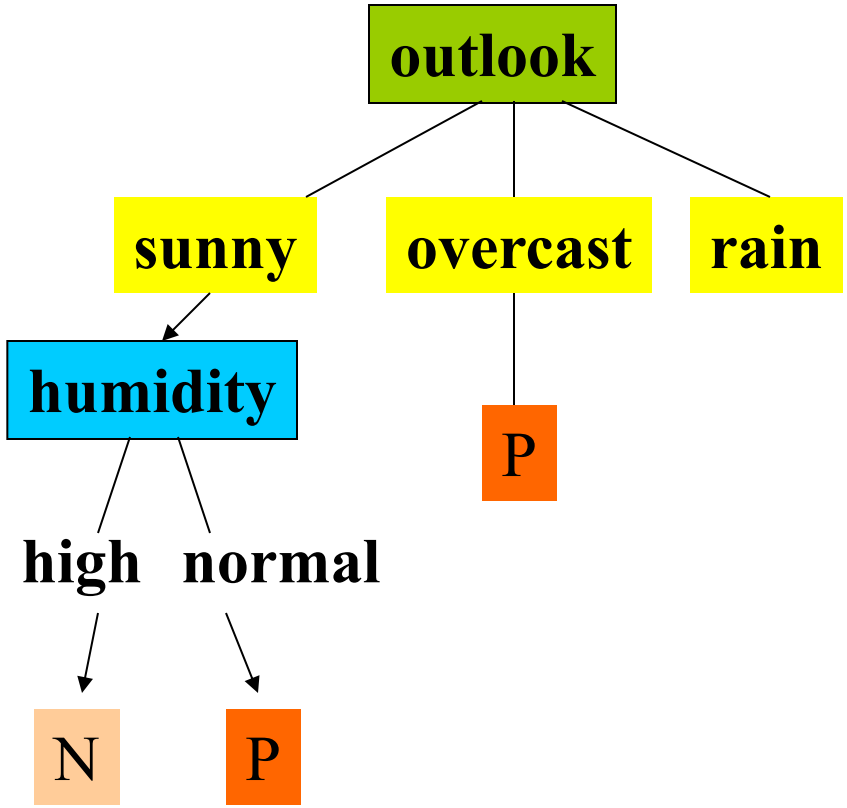
Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Decision Tree Construction



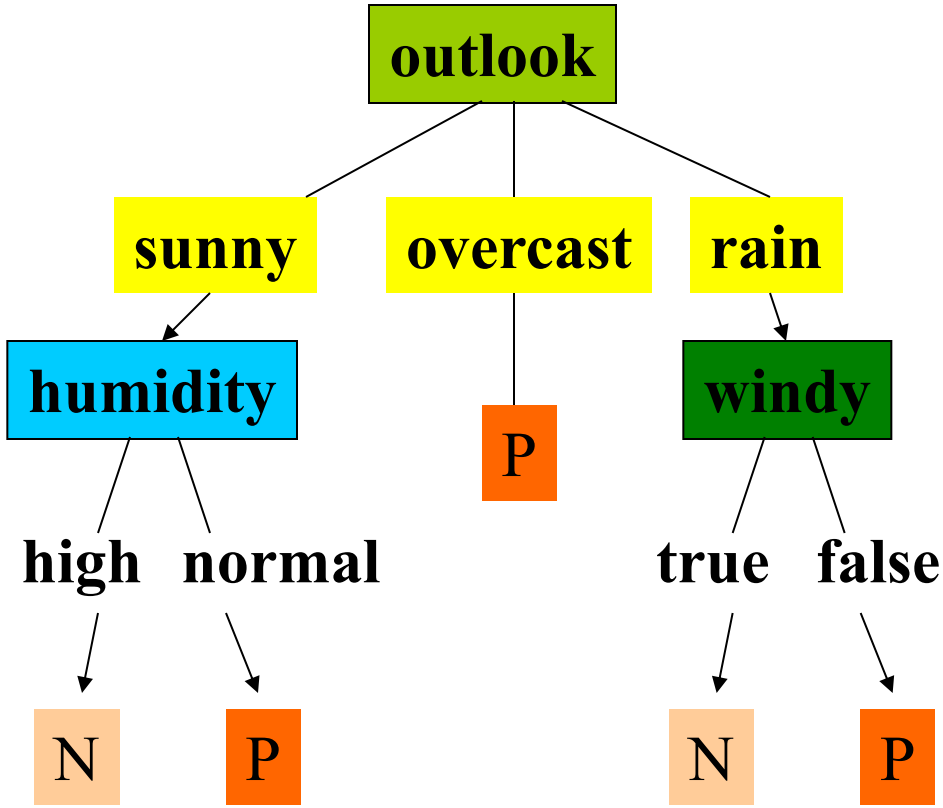
Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Decision Tree Construction



Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

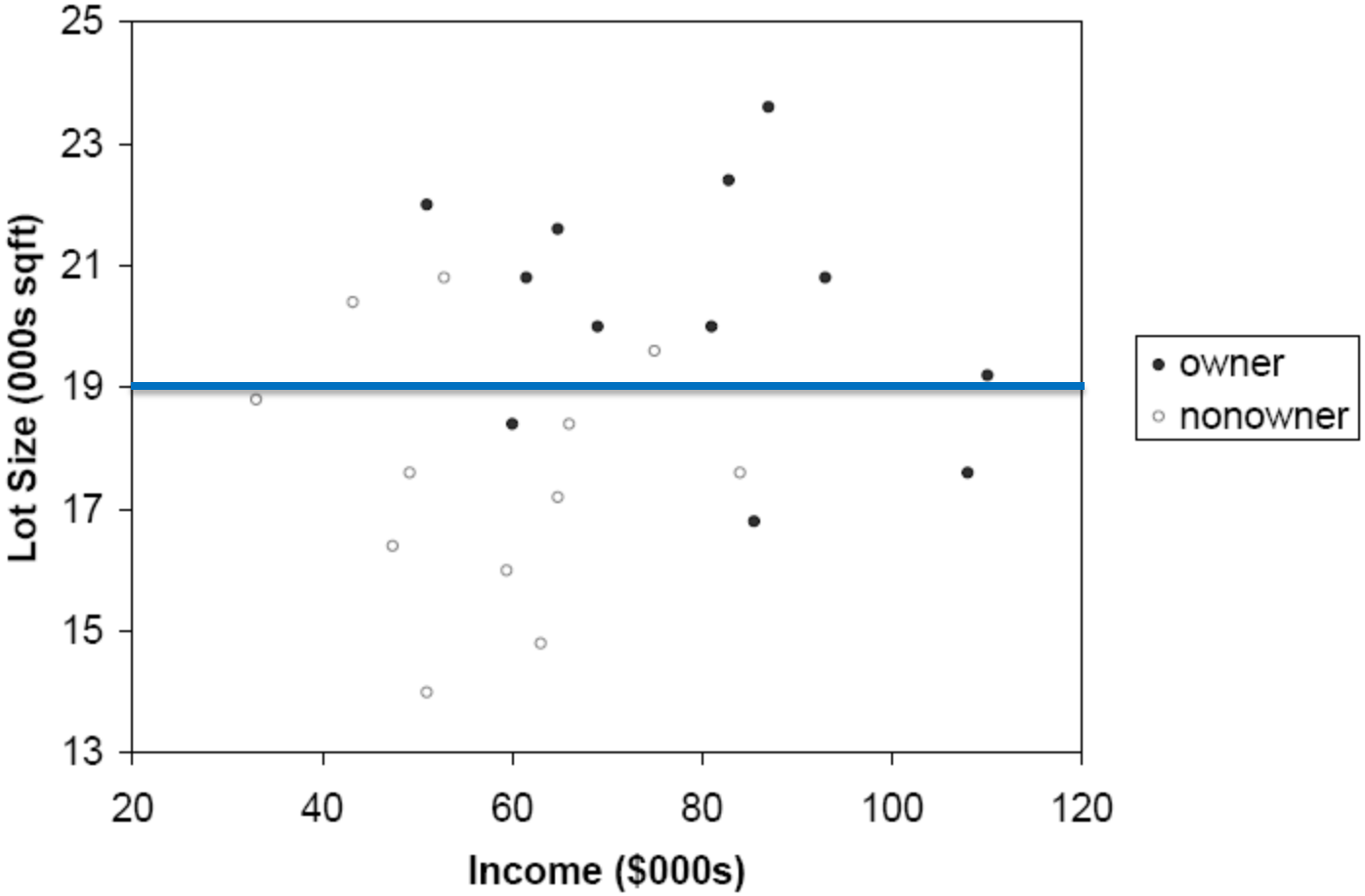
Decision Tree Construction



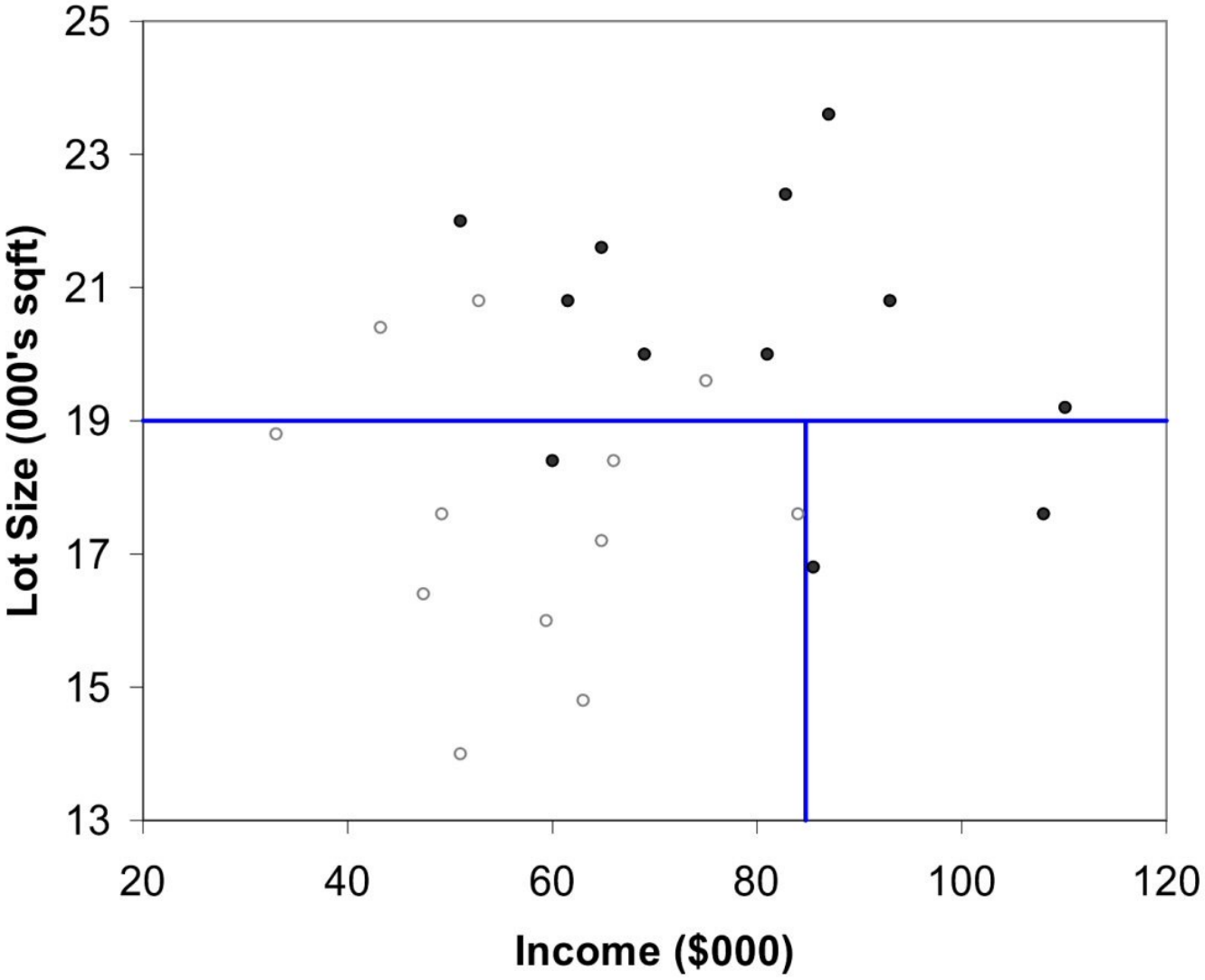
Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Income	Lot_Size	Ownership
60.0	18.4	owner
85.5	16.8	owner
64.8	21.6	owner
61.5	20.8	owner
87.0	23.6	owner
110.1	19.2	owner
108.0	17.6	owner
82.8	22.4	owner
69.0	20.0	owner
93.0	20.8	owner
51.0	22.0	owner
81.0	20.0	owner
75.0	19.6	non-owner
52.8	20.8	non-owner
64.8	17.2	non-owner
43.2	20.4	non-owner
84.0	17.6	non-owner
49.2	17.6	non-owner
59.4	16.0	non-owner
66.0	18.4	non-owner
47.4	16.4	non-owner
33.0	18.8	non-owner
51.0	14.0	non-owner
63.0	14.8	non-owner

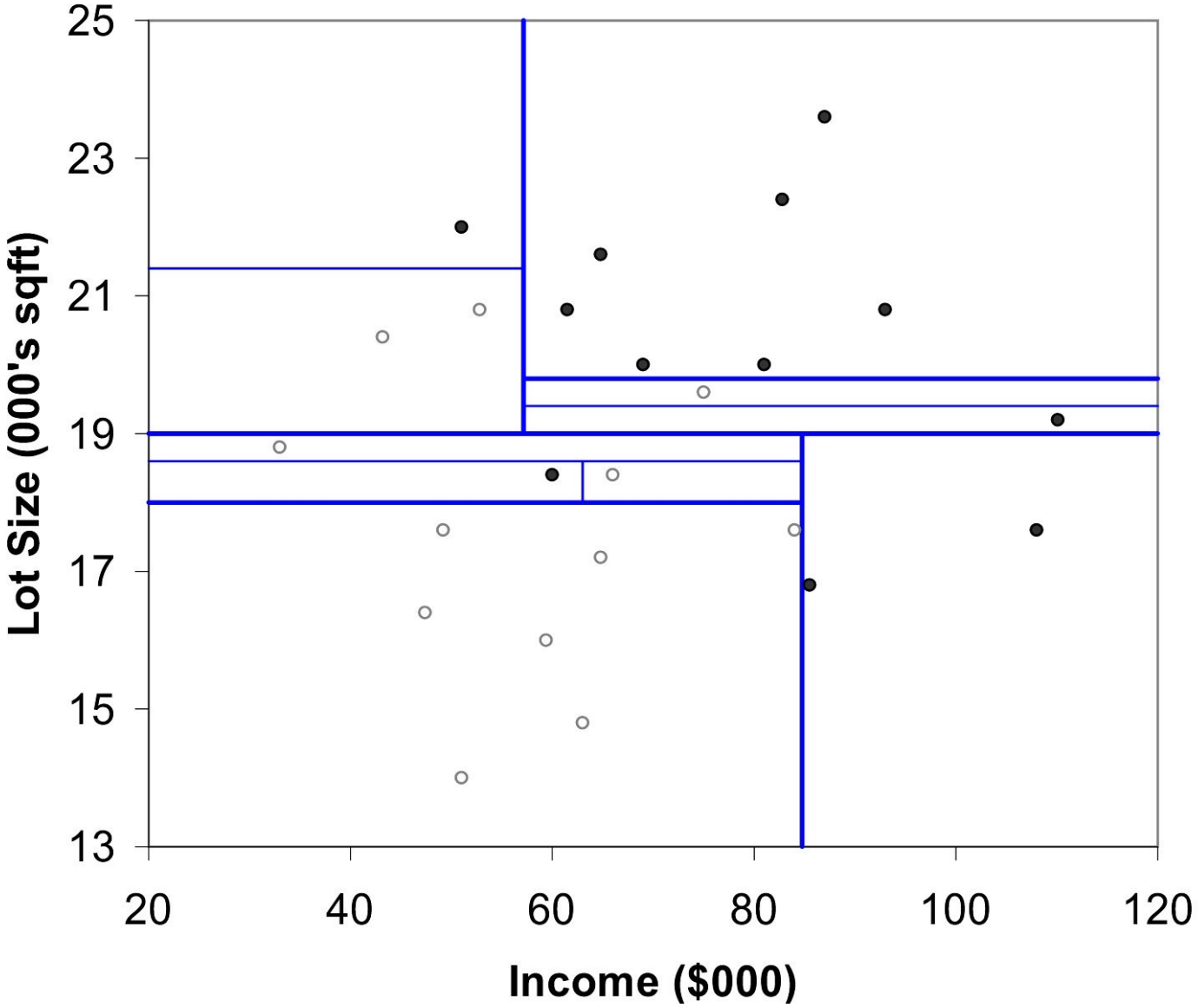
The first split: Lot Size = 19,000



Second Split: Income = $84,000



After All Splits



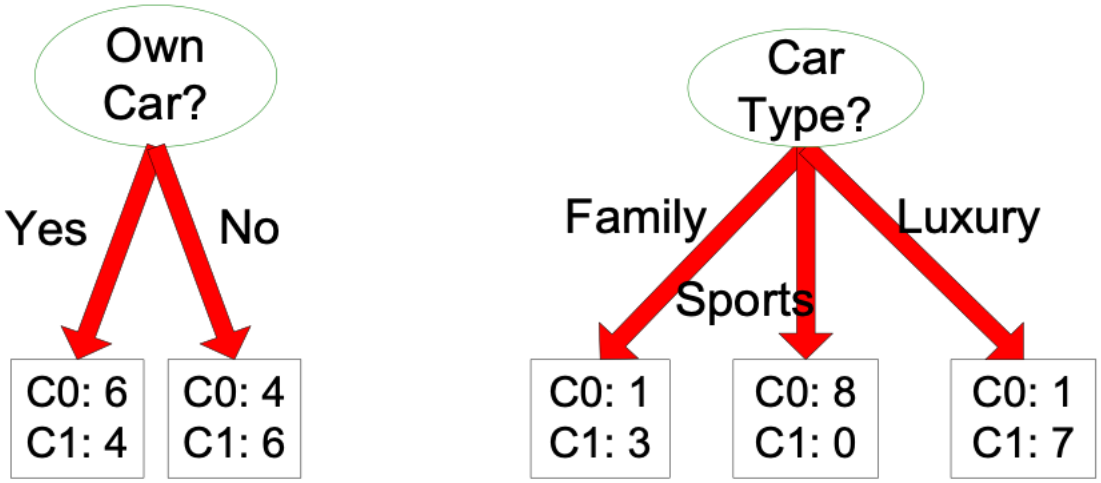
How to determine the Best Split

- Need a measure of node impurity:

C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous,	**Homogeneous (Pure),**
High degree of impurity	**Low degree of impurity**

How to determine the Best Split

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



Which test condition is the best?

Measures of Node Impurity

- Entropy
- Gini Index
- Misclassification error
- Etc.

Measure of Impurity: GINI

- Gini Index for a given node t :

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

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

- Maximum when records are equally distributed among all classes, implying least interesting information
- Minimum 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	

Examples for computing GINI

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

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

Highest GINI when equal distribution among all classes => 1-(1/N) where N is the number of classes.

Measure of Impurity: Entropy

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

Examples for computing Entropy

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

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) $\log_2$ (5/6) = 0.65

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

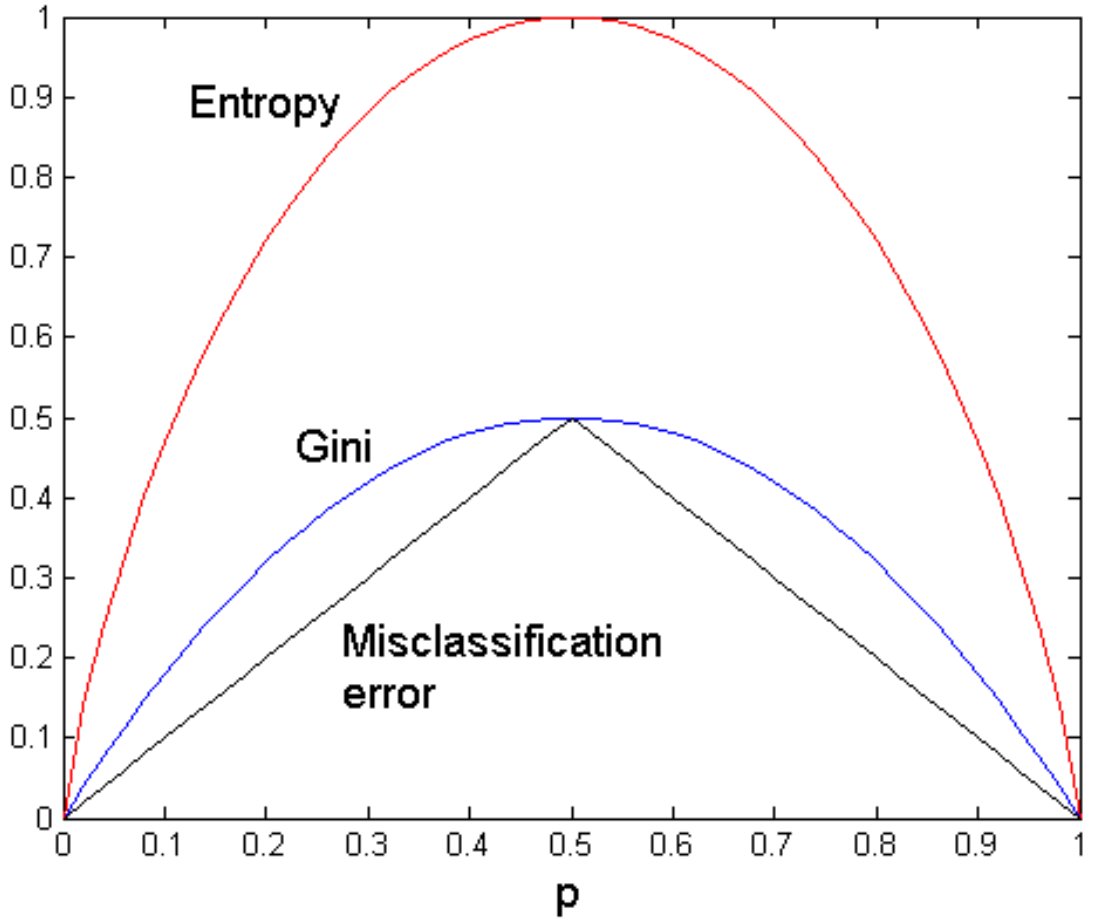
P(C1) = 2/6 P(C2) = 4/6

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

Highest Entropy when equal distribution among all classes

Comparison among Splitting Criteria

For a 2-class problem:



Popular Algorithms for Decision Tree Induction

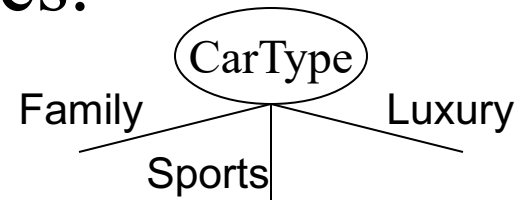
- Entropy and GINI produce similar tree structure.
- Entropy-based:
  - ID3, C4.5
  - Large number of partitions where each portion having small number of records.
  - Better of Binary Trees.
- GINI-based:
  - CART (Classification and Regression Tree)
  - Small number of partitions where each portion having large number of records.
  - Better for N-ary Trees.
- More Scalable Algorithms:
  - SLIQ, SPRINT
  - Etc.

How to Specify Test Condition?

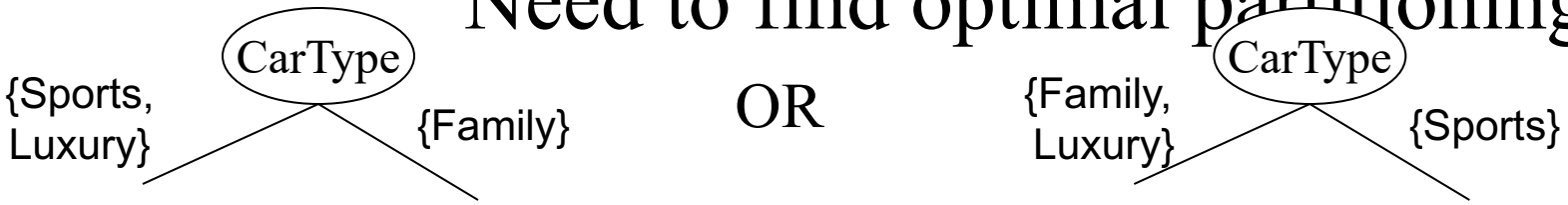
- Depends on attribute types
  - Nominal
  - Continuous
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Split on Categorical Variables

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



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



Split on Categorical Variables

- Examine all possible ways in which the categories can be split.

- E.g., categories A, B, C can be split 3 ways

  {A} and {B, C}

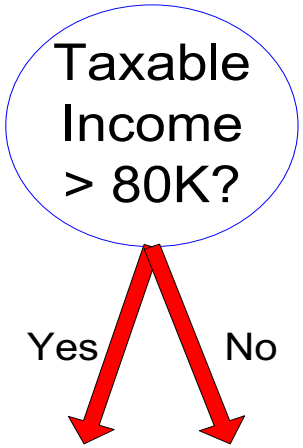
  {B} and {A, C}

  {C} and {A, B}

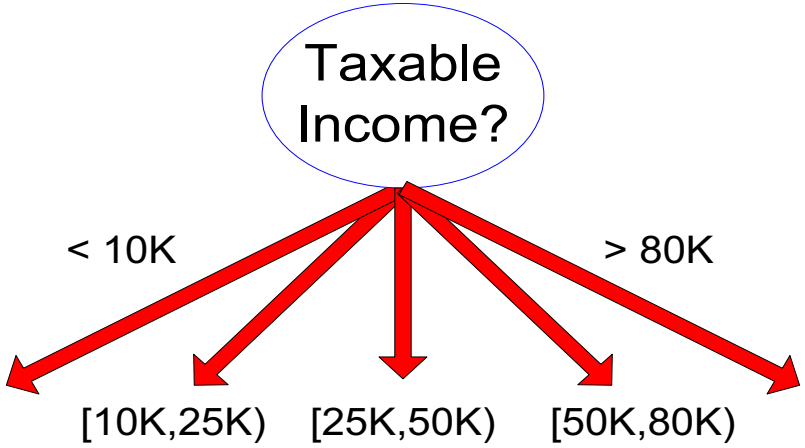
- With many categories, # of splits becomes huge
- Some supports only binary categorical variables

Splitting Based on Continuous Attributes

- Different ways of handling
  - Discretization to form an ordinal categorical attribute
  - Binary Decision: (A < v) or (A ≥ v)
    - Consider all possible splits and finds the best cut
    - Can be more compute intensive



(i) Binary split



(ii) Multi-way split

Impurity and Recursive Partitioning

- Obtain overall impurity measure (**weighted avg. of individual nodes**)
- At each successive stage, compare this measure across all possible splits in all variables
- Choose the split that **reduces impurity the most**
- Chosen split points become nodes on the tree

Splitting Based on Entropy...

- Information Gain:

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

Parent Node, p is split into k partitions;

n_i is number of records in partition i

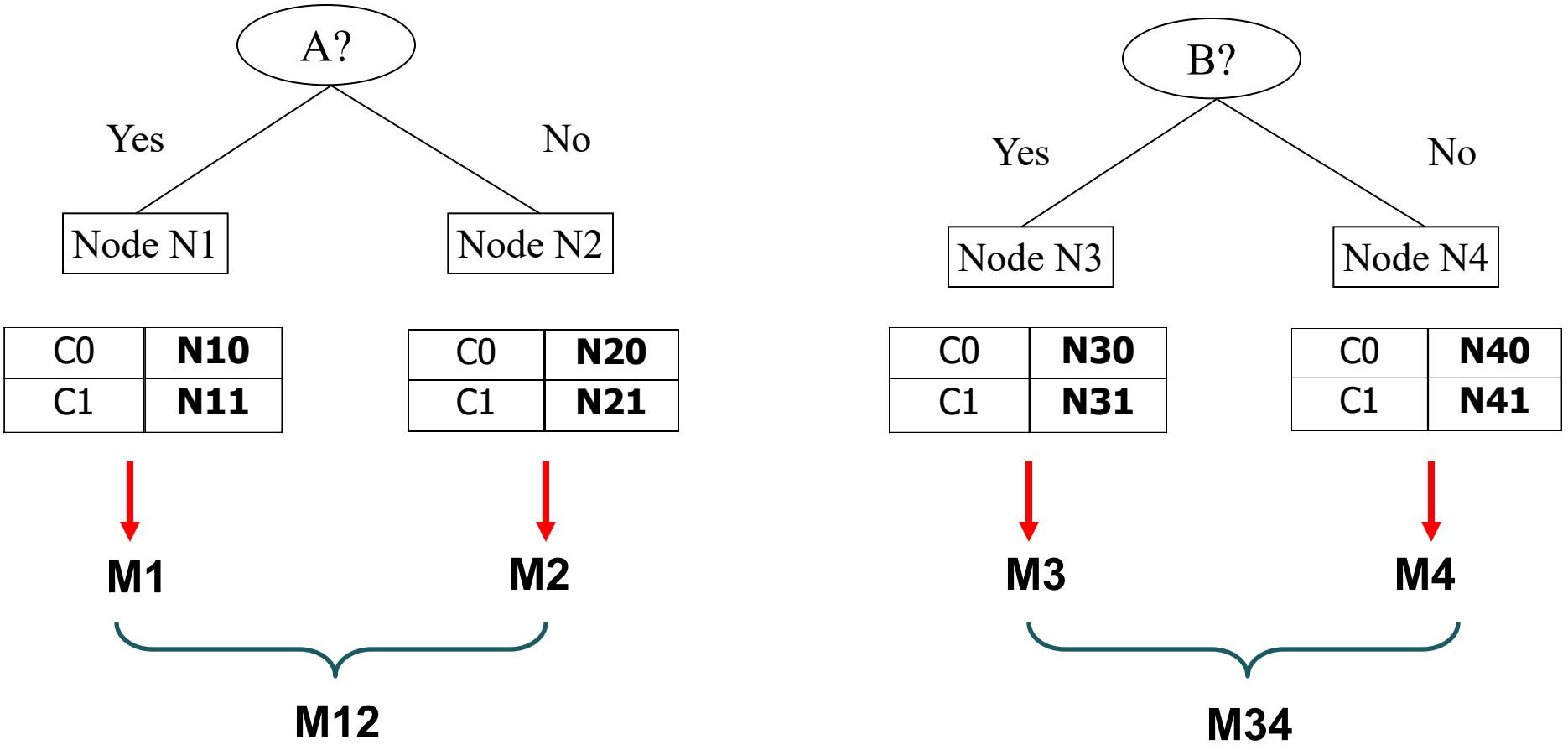
- GAIN = Entropy(Parent Node) - Entropy(Child Nodes)
- Measures Reduction in Entropy achieved because of the split.
- Choose the split that achieves most reduction (maximizes GAIN)
- Used in C4.5

How to Find the Best Split

Before Splitting:

C0	**N00**
C1	**N01**

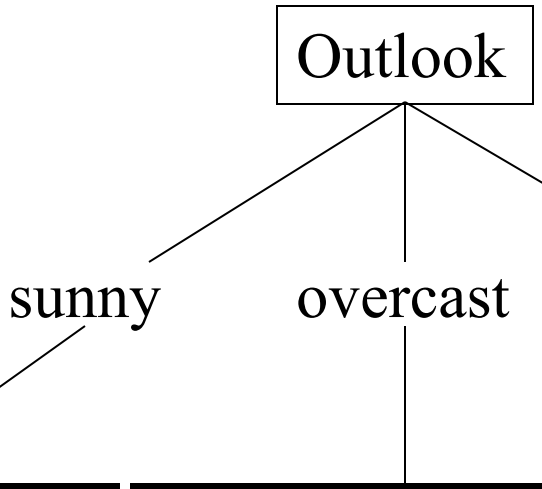
→ **M0**



Information-Gain = M0 – M12 vs M0 – M34

Information-Gain(Outlook) ?

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N



1	Hot	High	False	N
2	Hot	High	True	N
8	Mild	High	False	N
9	Cool	Normal	False	P
11	Mild	Normal	True	P

3	Hot	High	False	P
7	Cool	Normal	true	P
12	Mild	High	True	P
13	Hot	Normal	False	P

4	Mild	High	False	P
5	Cool	Normal	False	P
6	Cool	Normal	True	N
10	Mild	Normal	False	P
14	Mild	High	True	N

Information-Gain(Outlook) ?

P	**9**
N	**5**

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

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

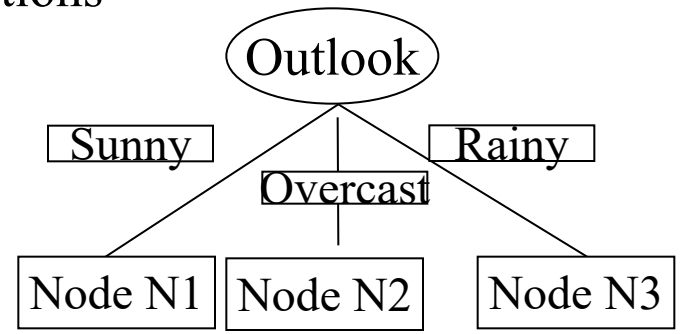
P(P) = 9/14 P(N) = 5/14

Entropy(Dataset) = – (9/14) $\log_2$ (9/14) – (5/14) $\log_2$ (5/14)

= 0.41 + 0.54 = 0.94

Information-Gain (Outlook)

- Splits into 3 partitions



	Parent
P	9
N	5
Entropy = 0.94	

	N1	N2	N3
P	2	4	3
N	3	0	2
Entropy=0.69			

Entropy(Sunny)
= – (2/5) $\log_2$ (2/5) – (3/5) $\log_2$ (3/5)
= 0.971

Entropy(Overcast)
= – (4/4) $\log_2$ (4/4)
= 0

Entropy(Rainy)
= – (3/5) $\log_2$ (3/5) – (2/5) $\log_2$ (2/5)
= 0.971

Entropy(Outlook)
= 5/14 * E(Sunny) + 4/14 * E(Overcast) + 5/14 * E(Rainy)
= 5/14 * 0.971 + 4/14 * 0 + 5/14 * 0.971

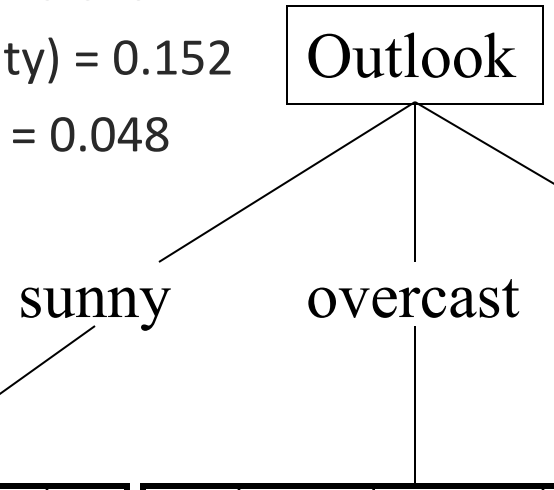
If we split using Outlook, Impurity is reduced to =>

Information-Gain(Outlook) = 0.94 - 0.69 = 0.247
= 0.25

Outlook has the highest gain

- Info Gain(outlook) = 0.247
- Info Gain(temp) = 0.029
- Info Gain(humidity) = 0.152
- Info Gain(windy) = 0.048

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N



1	Hot	High	False	N
2	Hot	High	True	N
8	Mild	High	False	N
9	Cool	Normal	False	P
11	Mild	Normal	True	P

3	Hot	High	False	P
7	Cool	Normal	true	P
12	Mild	High	True	P
13	Hot	Normal	False	P

4	Mild	High	False	P
5	Cool	Normal	False	P
6	Cool	Normal	True	N
10	Mild	Normal	False	P
14	Mild	High	True	N

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,

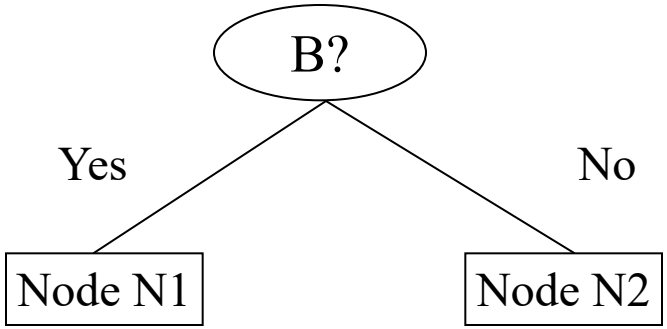
$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

where, n_i = number of records at child i,

n = number of records at node p.

Binary Attributes: Computing GINI Index

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



	Parent
C1	6
C2	6
Gini = 0.500	

Gini(N1)
= 1 – (5/7)² – (2/7)²
= 0.194

Gini(N2)
= 1 – (1/5)² – (4/5)²
= 0.528

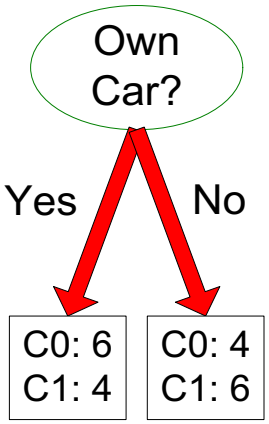
	N1	N2
C1	5	1
C2	2	4
Gini=0.333		

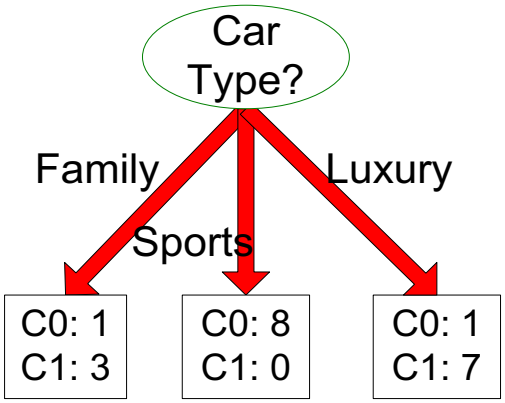
Gini(Children)
= 7/12 * 0.194 +
5/12 * 0.528
= 0.333

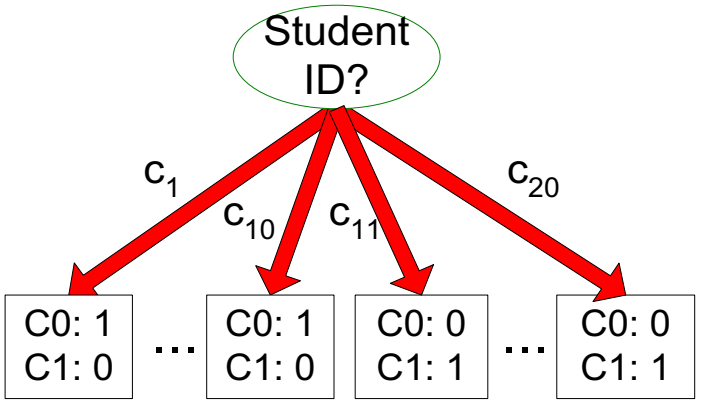
Gain = 0.5 - 0.333

More on Information-Gain => Gain-Ratio

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







Which test condition is the best?

More on Information-Gain => Gain-Ratio

- The Information-gain with a large number of child nodes tends to be higher than the Information-gain with a less number of children.
- Solution is to use **Gain-Ratio**.

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

- p(j) is the proportion of the number of data of child nodes.
- m is the number of child nodes.

- The **SplitInfo** value will act as Normalization Factor, which will increase with the number of child nodes.
- The SplitInfo value will have a maximum value equal to log(mm) when the proportion of the data of all child nodes is equal, 1/m.
- Therefore, the **Gain-Ratio** of the condition with the number many child nodes will be scaled down with a large **SplitInfo** value based on the number of child nodes.

Decision Tree Induction

- Popular Algorithms:
  - Hunt's Algorithm, CART
  - ID3, C4.5
- Scalable Algorithms:
  - SLIQ,
  - SPRINT
  - Etc.

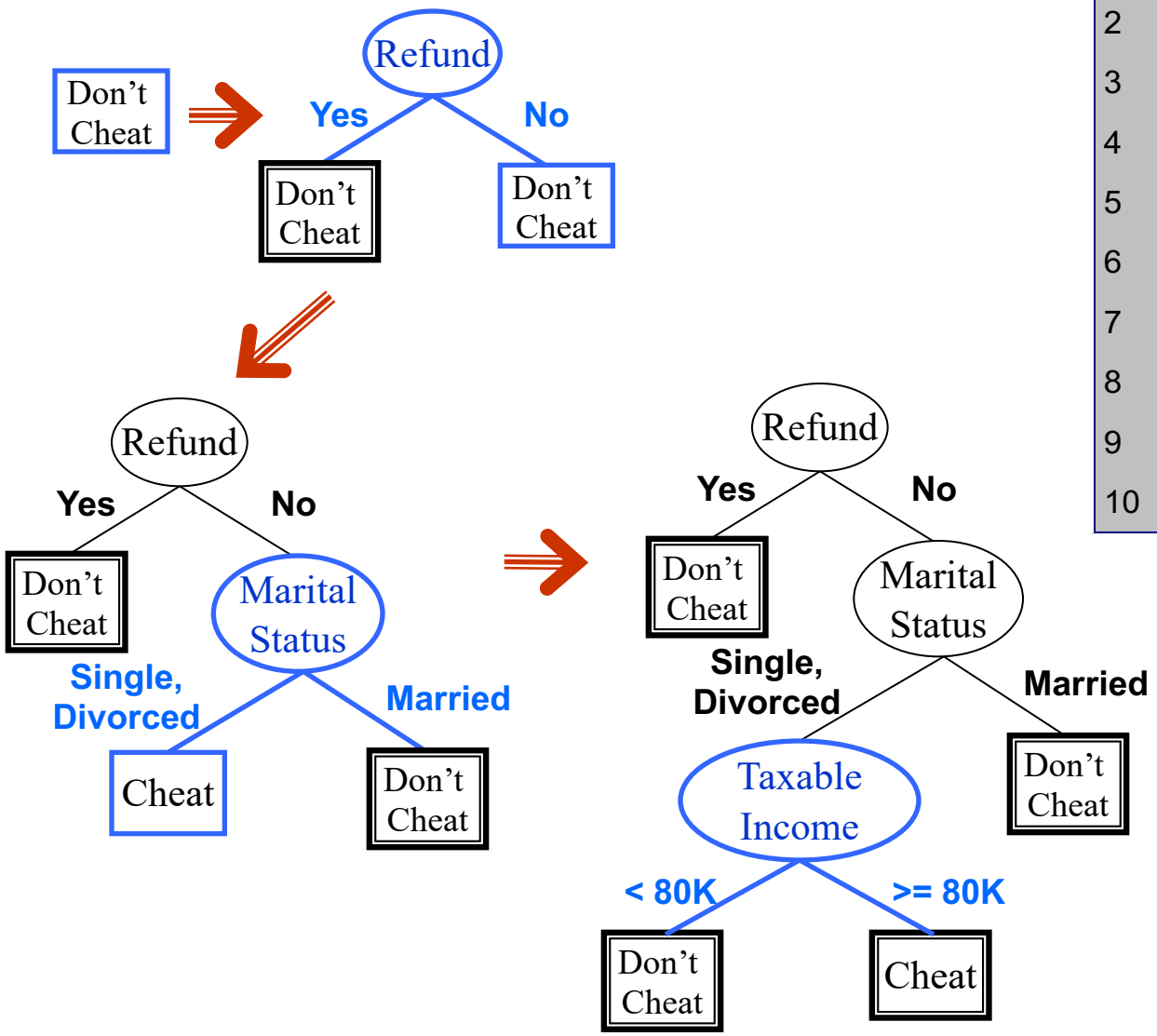
Embedded Feature Selection

Reasons for selecting only a subset of attributes:

- Determining the most informative attributes.
- Better insights and business understanding
- Better explanations and more tractable models
- Reduced cost
- Faster predictions
- Better predictions!
  - Over-fitting

Example of Tree Construction

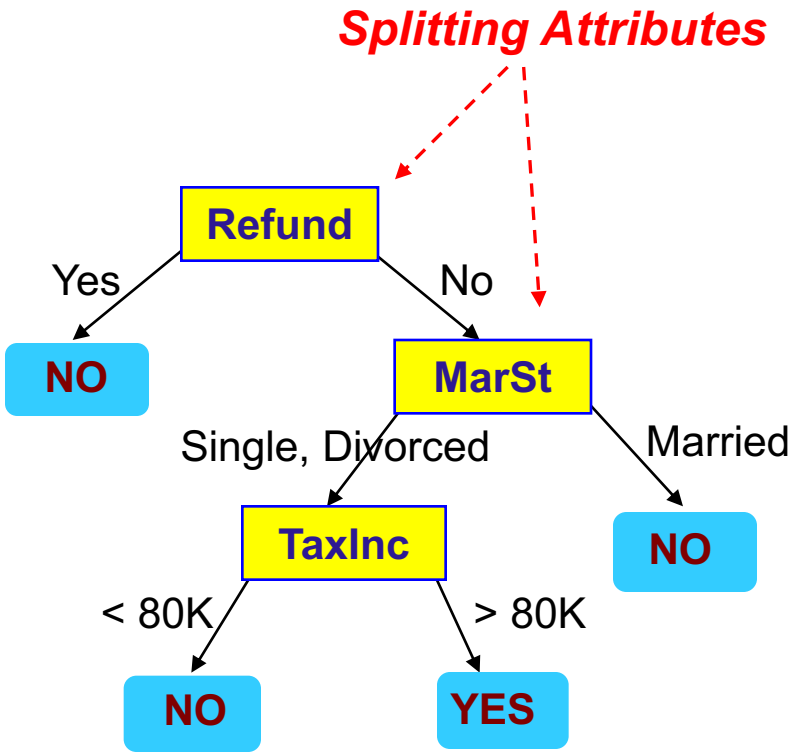
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Example of a Decision Tree

Tid	Refund (categorical)	Marital Status (categorical)	Taxable Income (continuous)	Cheat (class)
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

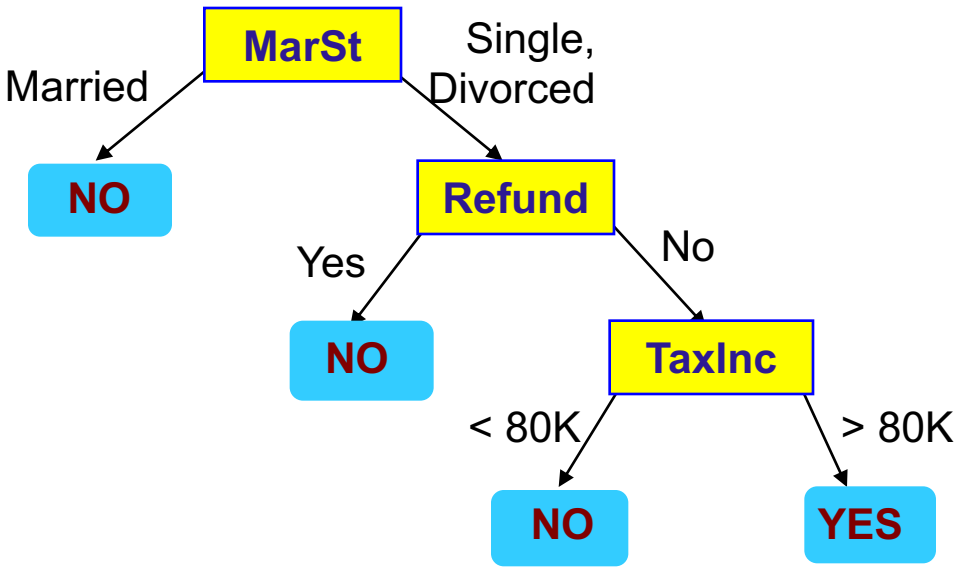
Training Data



Model: Decision Tree

Another Example of Decision Tree

Tid	Refund (categorical)	Marital Status (categorical)	Taxable Income (continuous)	Cheat (class)
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

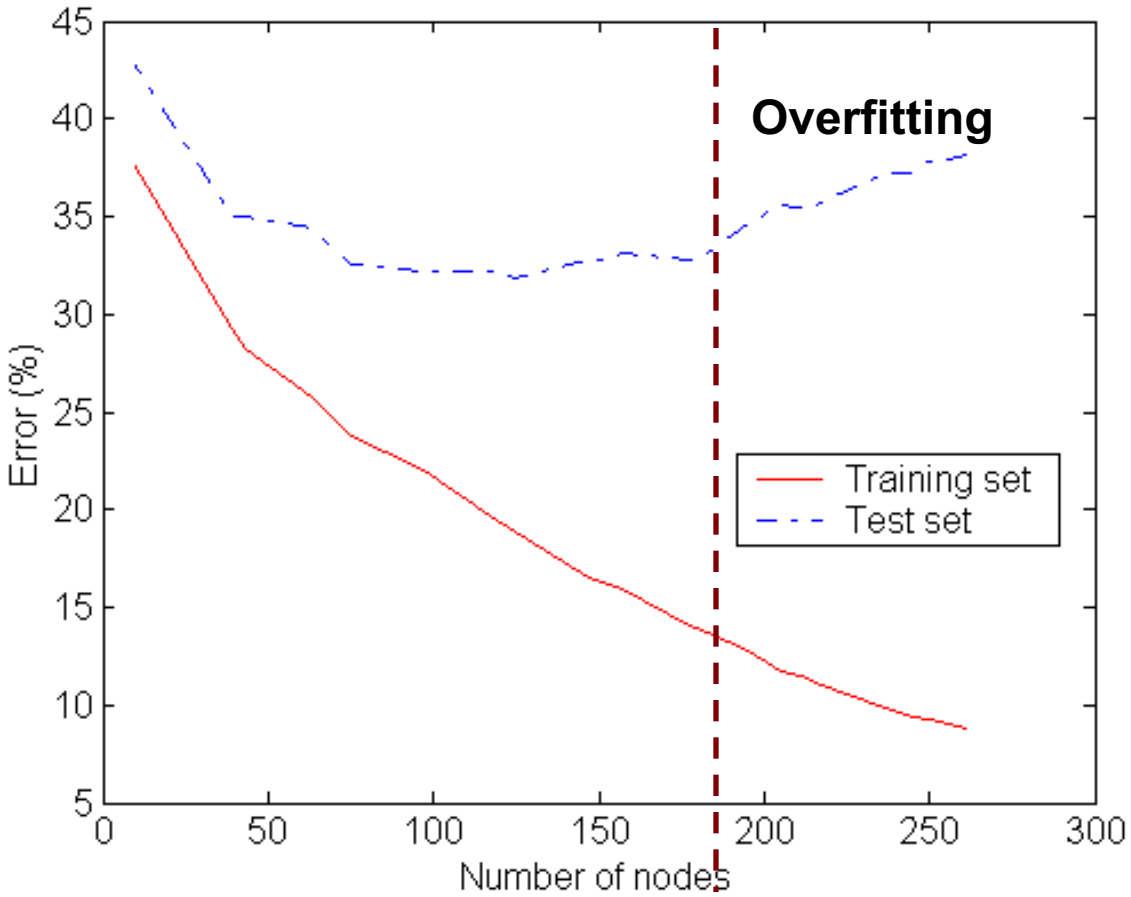


There could be more than one tree that fits the same data!

Decision Tree

- The branches always “point downwards”
- Every example always ends up at a leaf node with some specific class determination
  - Probability estimation trees, regression trees
- How do we create a classification tree from data?
  - Divide-and-conquer approach
  - Take each data subset and *recursively* apply attribute selection to find the best attribute to partition it
- When do we stop?
  - The nodes are pure,
  - There are no more variables, or
  - Even earlier

Underfitting and Overfitting



Underfitting: when model is too simple, both training and test errors are large

Overfitting: when the test error increases while the training error steadily decreases

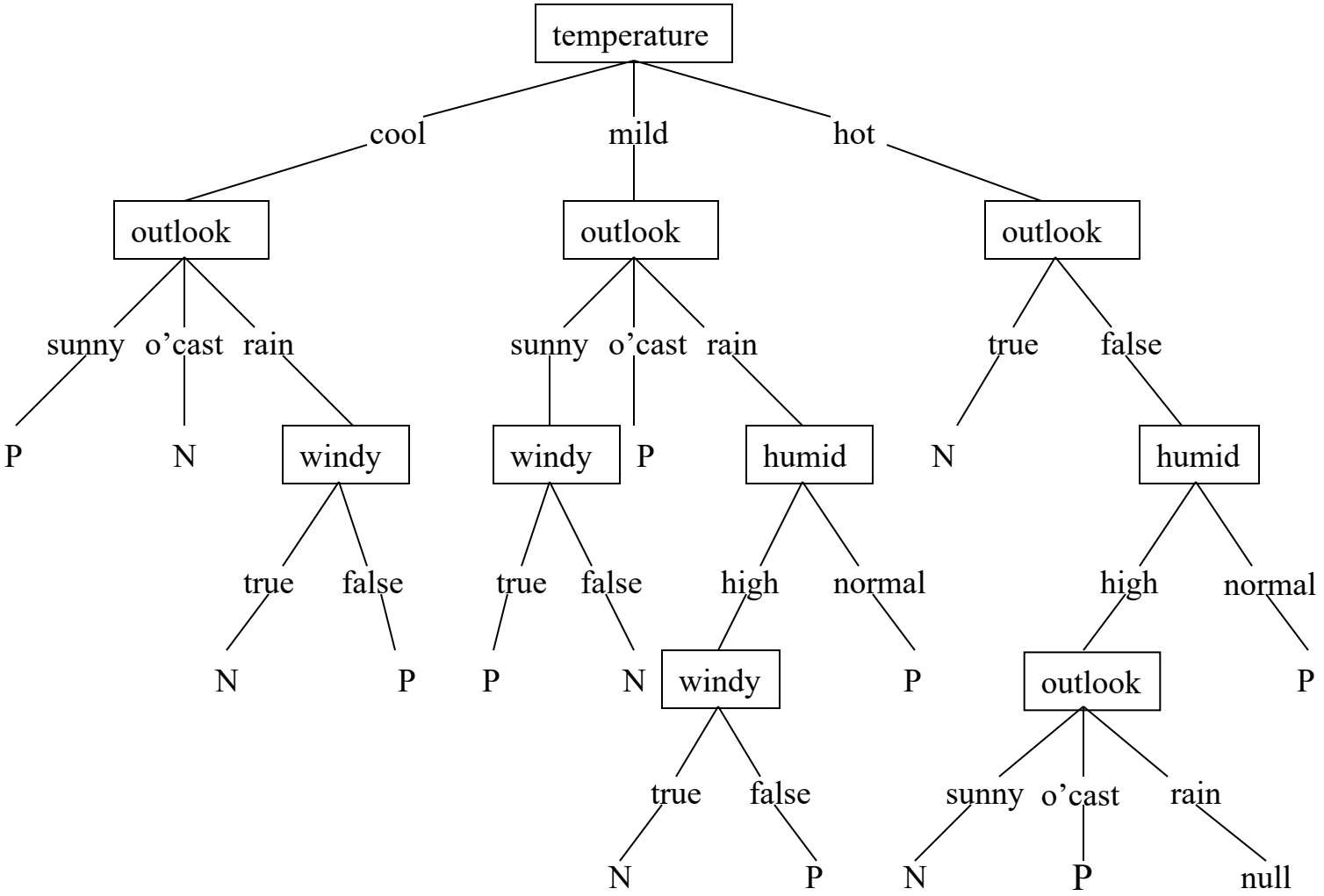
The size of the tree will become same as the size of the training data => too specific to the training => Overfitting => Cannot generalize very well

Key Ideas: Avoid Overfitting

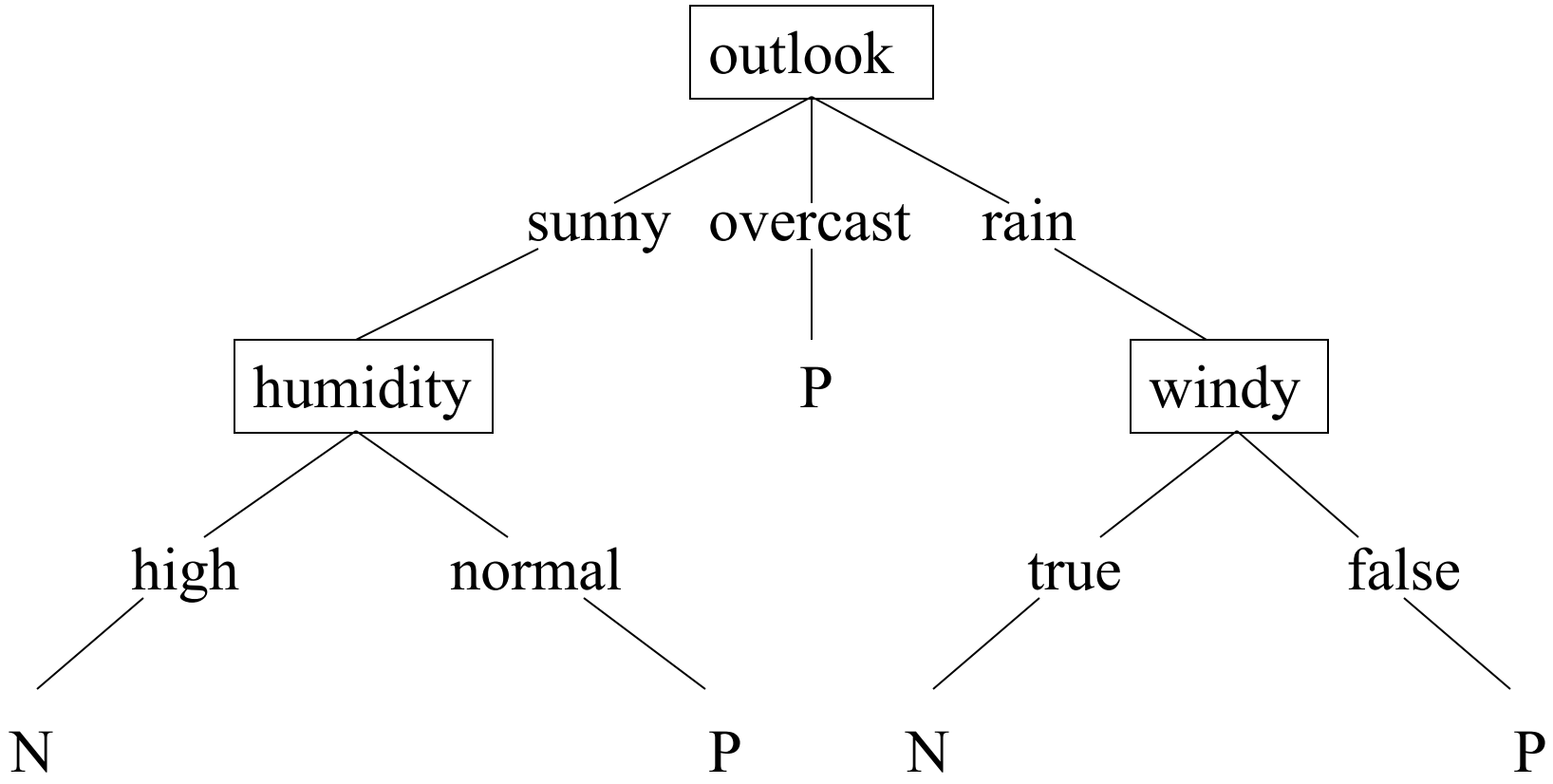
Recursive partitioning: Repeatedly split the records into two or more parts so as to achieve maximum homogeneity within the new parts

Pruning the tree: Simplify the tree by pruning peripheral branches to avoid overfitting

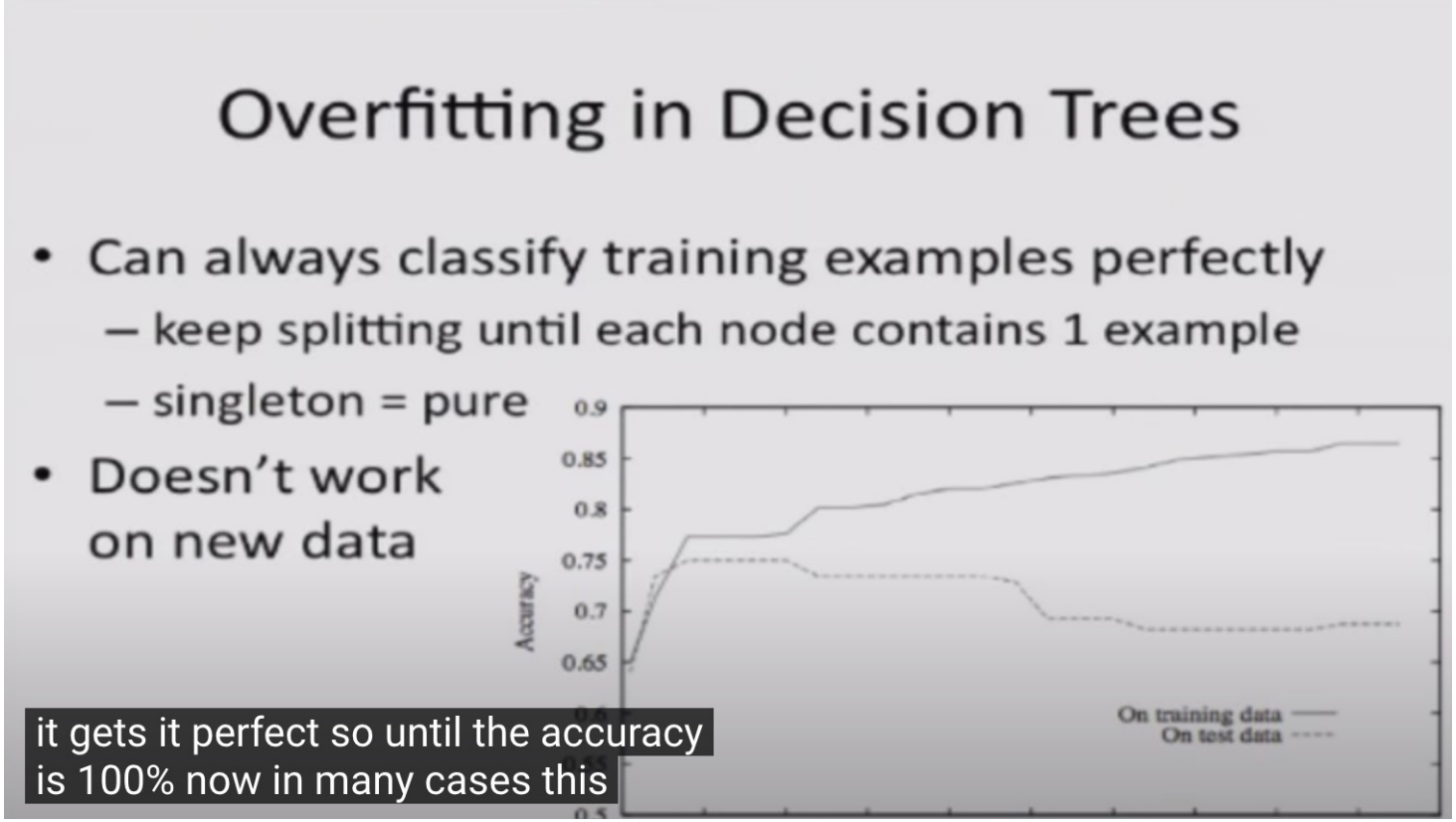
A Complex Decision Tree



A Simple Decision Tree



Overfitting



Underfitting: when model is too simple, both training and test errors are large

Overfitting: when the test error increases while the training error steadily decreases

The size of the tree will become same as the size of the training data => too specific to the training => Overfitting => Cannot generalize very well

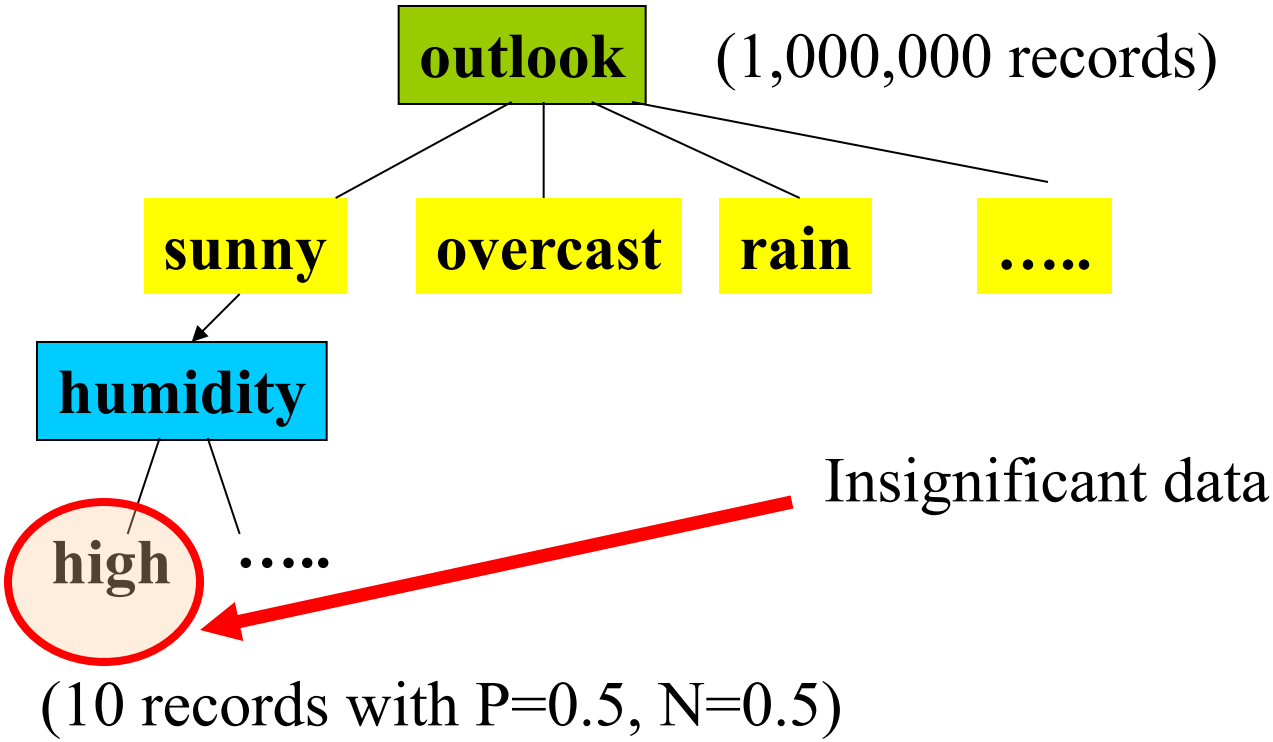
Stopping Tree Growth

- Natural end of process is 100% purity in each leaf
- This **overfits** the data, which end up fitting noise in the data
- Overfitting leads to low predictive accuracy of new data
- Past a certain point, the error rate for the validation data starts to increase

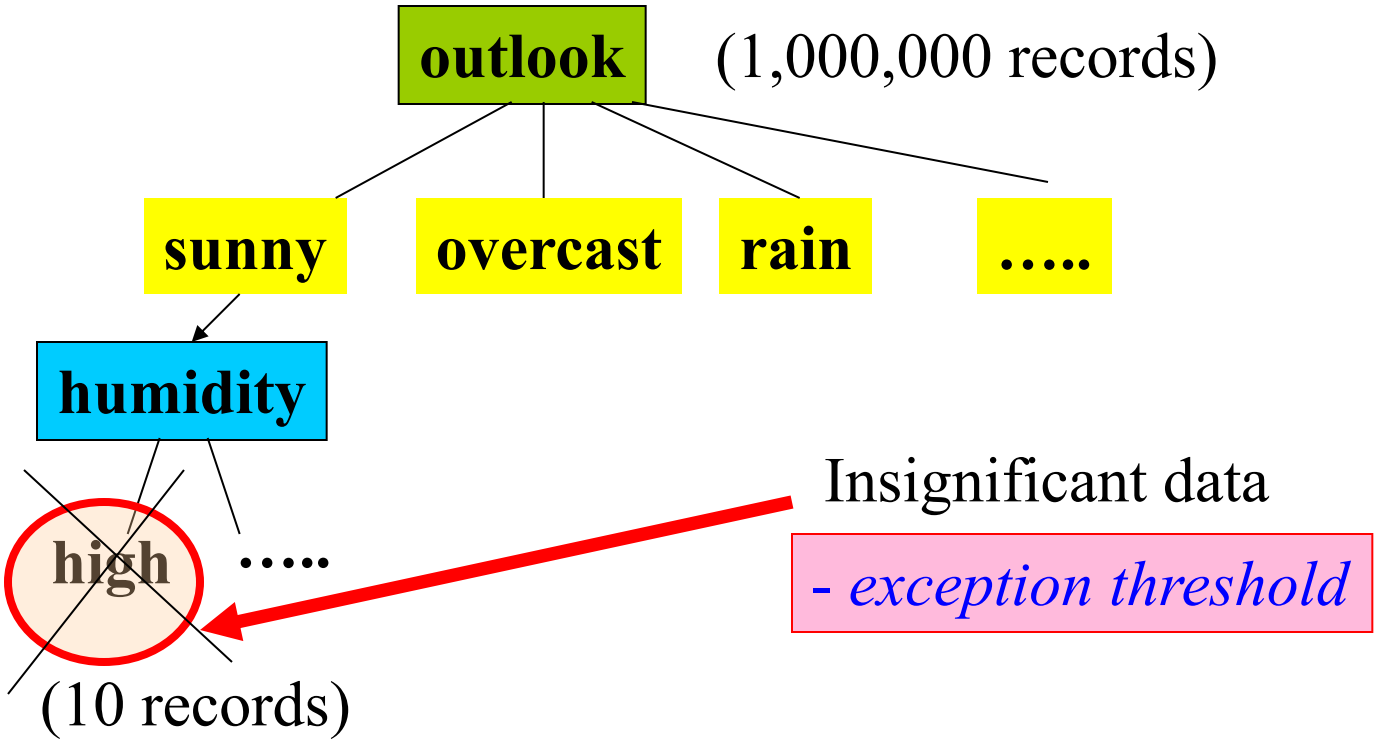
Pre-Pruning (Early Stopping)

- 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.
- Not necessary to get all data of the same class.
- Set the threshold of the impurity metrics in the current node (e.g. Entropy). If these metrics are lower than the Threshold value, it can stop creating child nodes.
- For example,
  - 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 χ^2 test).

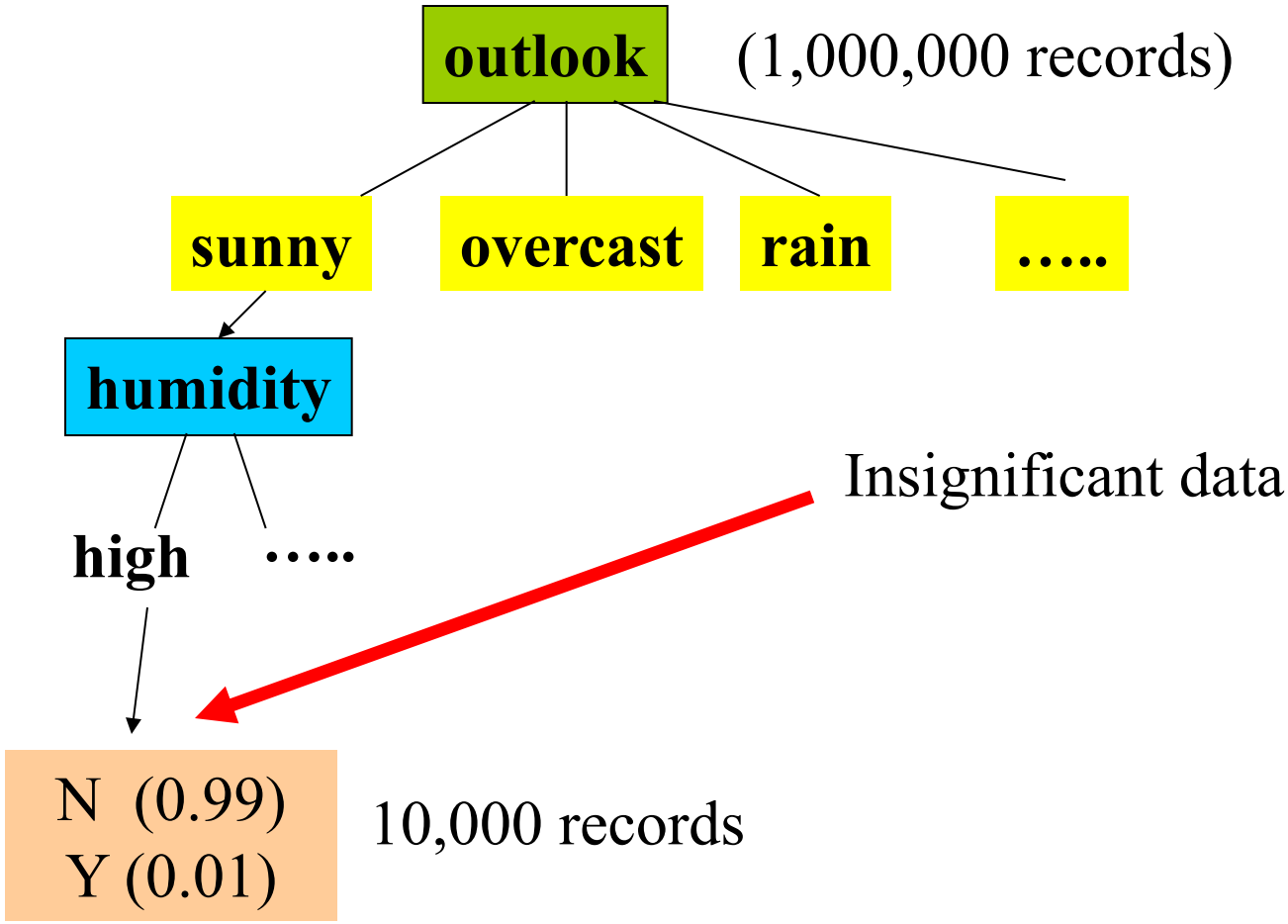
Stopping Tree Growth



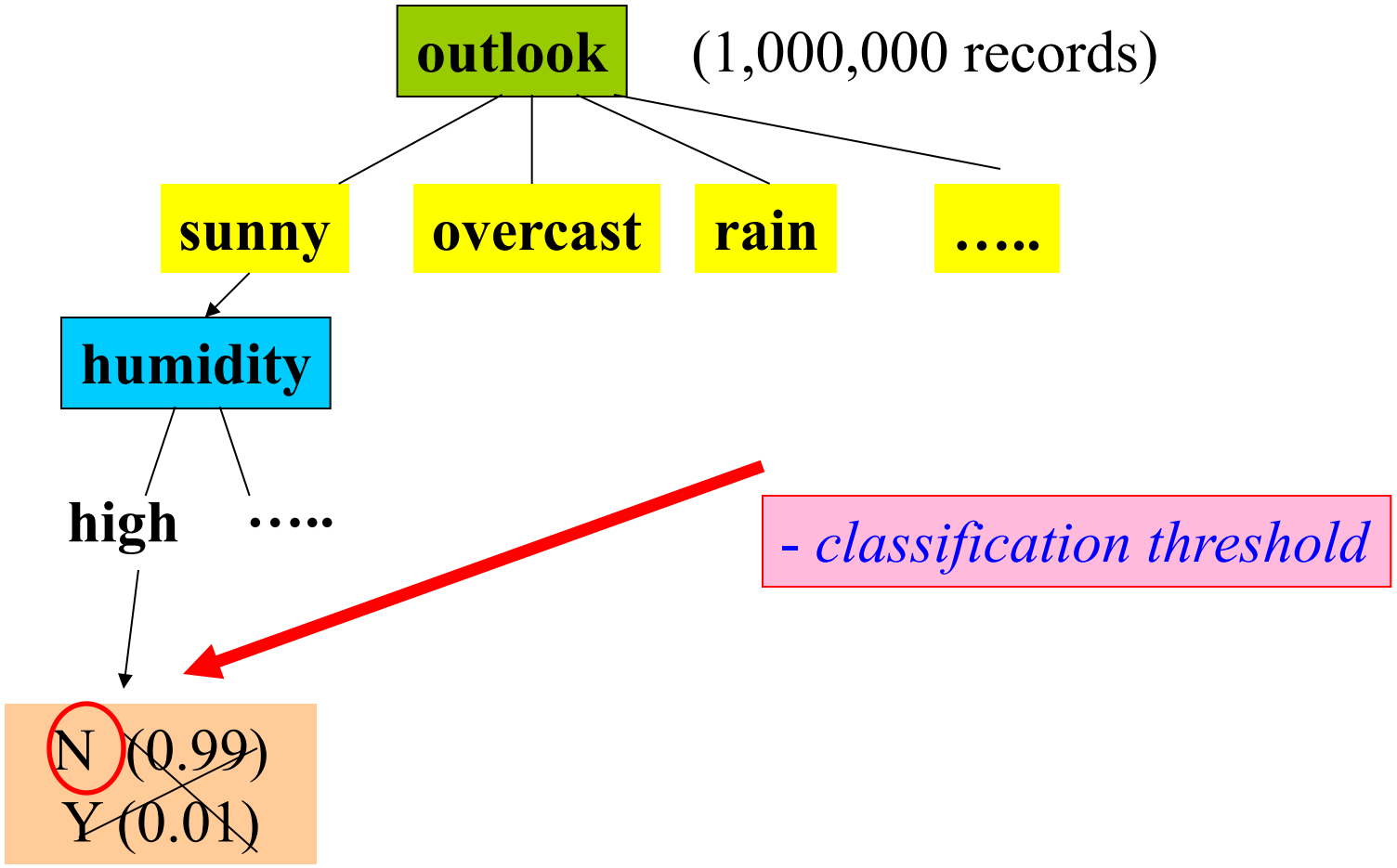
Stopping Tree Growth



Stopping Tree Growth



Stopping Tree Growth



Post-pruning

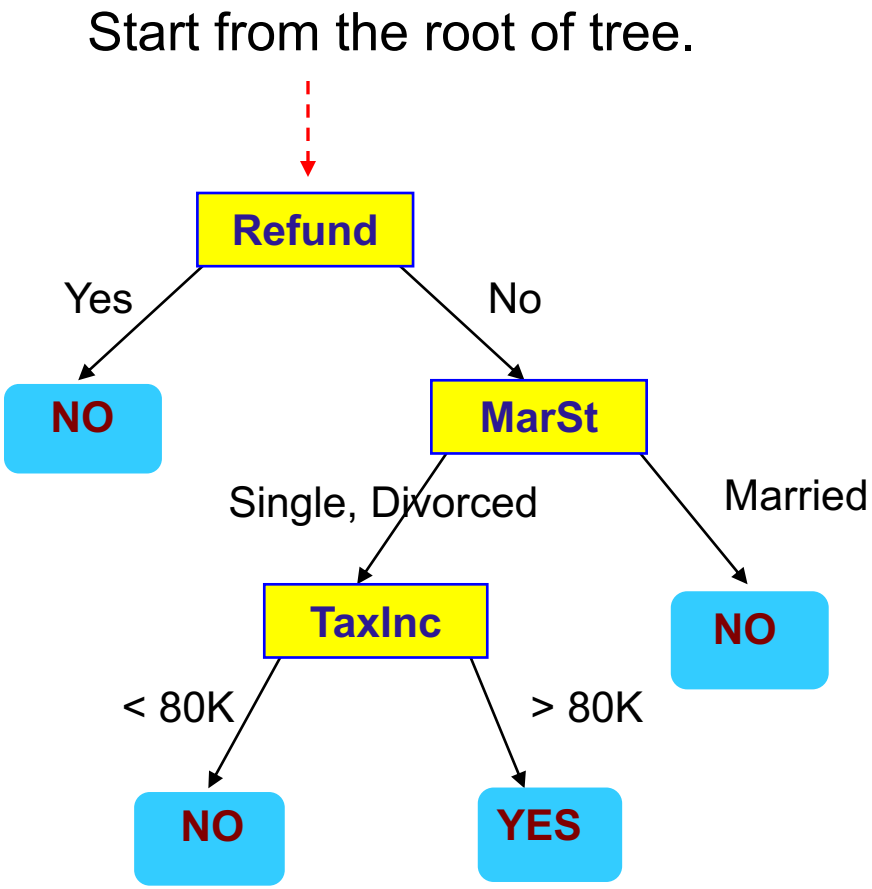
- Grow decision tree to its entirety, then cuts off nodes or subtrees to make the tree smaller.
- This method solves the problem that the tree is too specific causing the Overfitting Problem.
- Trim the nodes of the decision tree in a bottom-up fashion.
- For example,
  - Replace the entire subtree with leaf node. Provided that the impurity metric values (ex., Entropy) of this new node still do not exceed the Threshold value.
  - Pruning the branch that will not do well on the future data

using Validation Dataset.

How to use a tree ?

- Directly
  - Test the attribute value of unknown object against the tree.
  - A path is traced from root to a leaf which holds the label.
- Indirectly
  - Decision tree is converted to classification rules.
  - One rule is created for each path from the root to a leaf.
  - IF-THEN rules are easier for humans to understand.

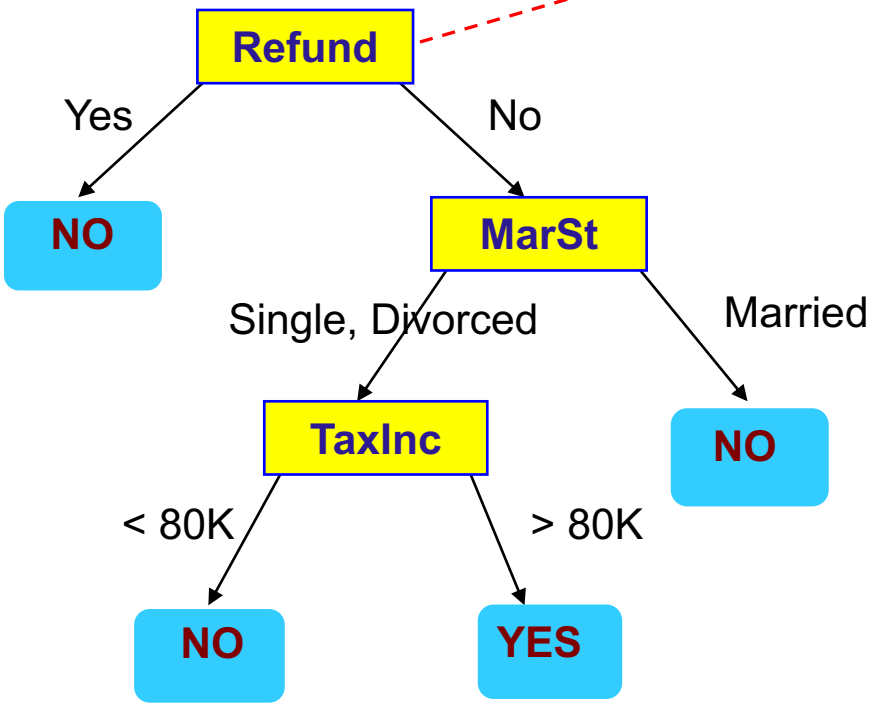
Apply Model to Test Data



Unseen Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

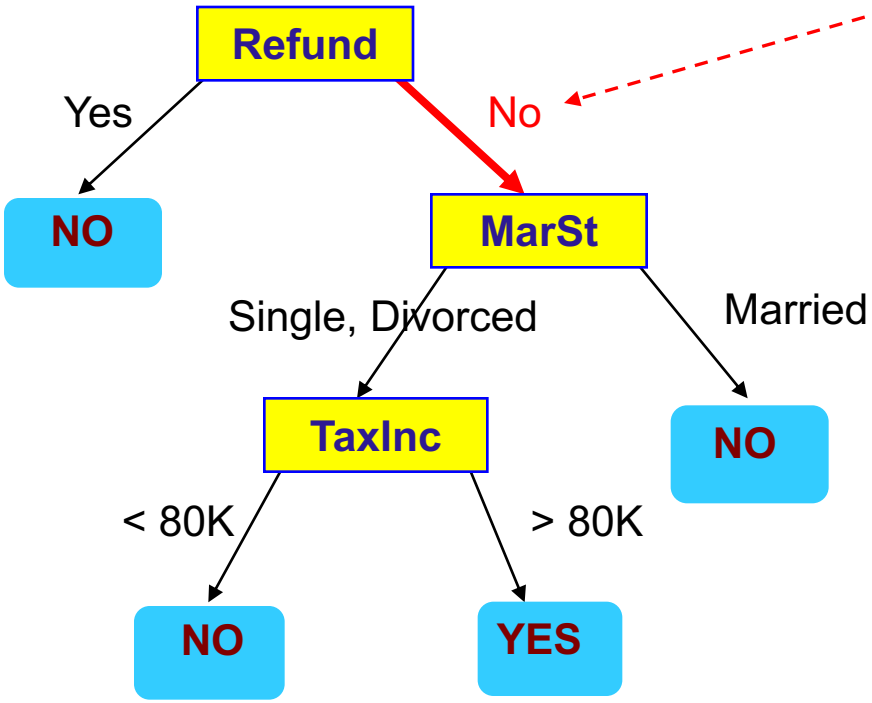
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



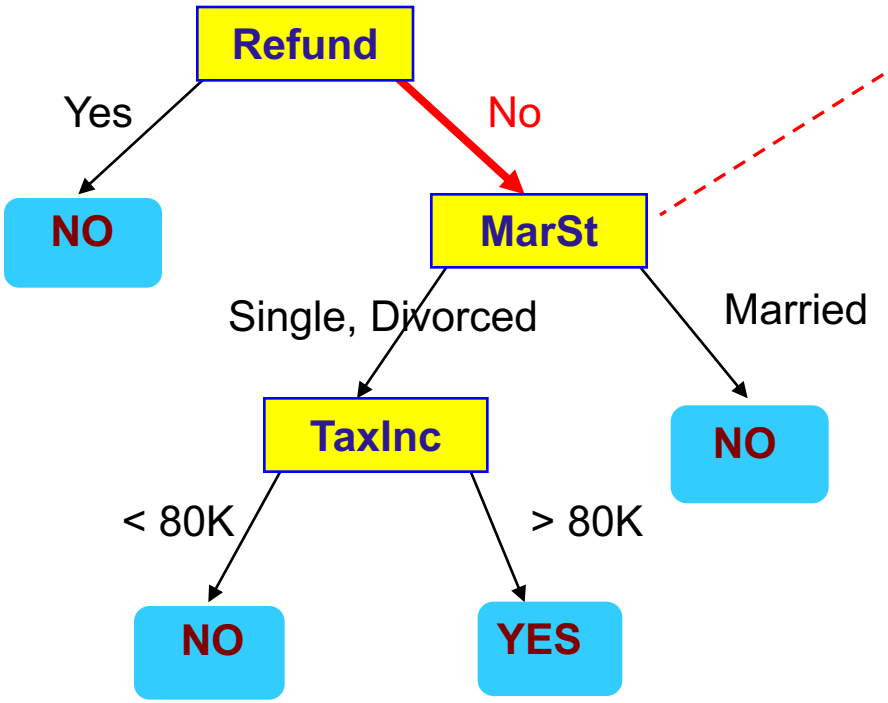
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

Test Data

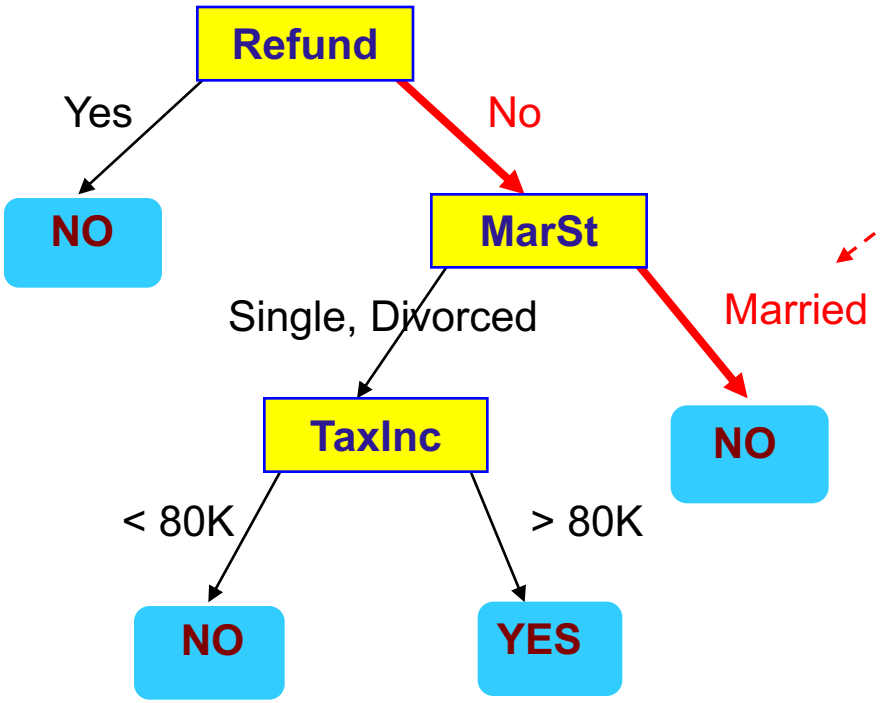
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



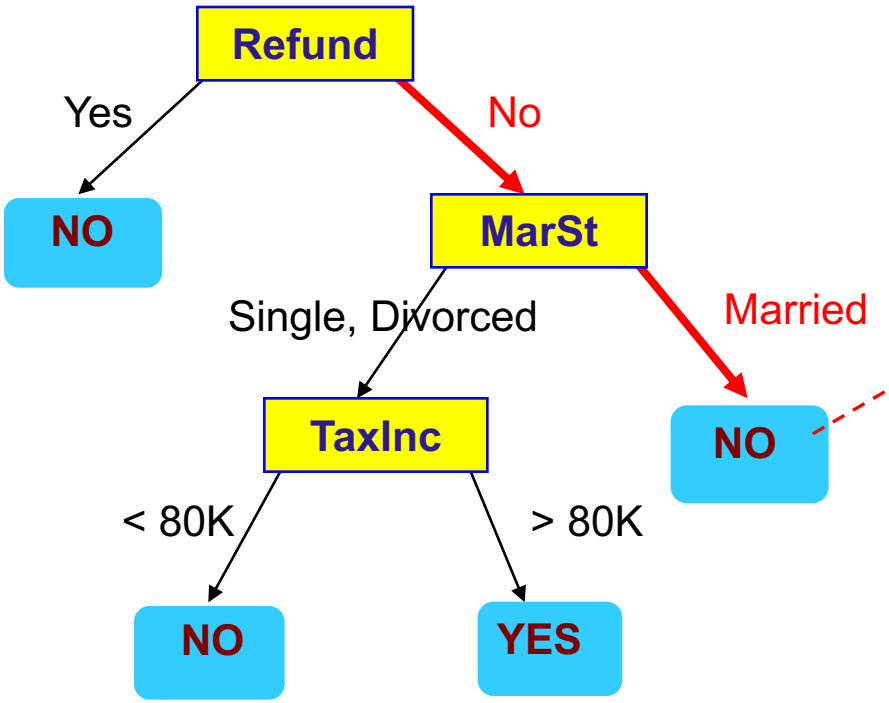
Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Assign Cheat to “No”

Decision Tree Based Classification

- Advantages:
  - Inexpensive to construct
  - Can work also for both classification and prediction
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees (after-pruning)
  - Accuracy is comparable to other classification techniques for many simple data sets
  - Produce rules that are easy to interpret & implement
  - Variable selection & reduction is automatic
  - Do not require the assumptions of statistical models.

Decision Tree Based Classification

- Dis-Advantages:
  - Problem with Big size tree.
  - Greedy => May not find the best tree.
  - Works best for data that Decision Boundary can be generated from Linear Hyperplanes.
  - Not suitable for any shape of Non Linear Decision Boundary.
  - If the learning data contains noise, it will cause overfitting problems if construct a tree that is too specific (Fully Grown Tree).

Random decision forest

- K different decision trees
  - Pick a random subset S, of training samples
  - For each subset => grow a full tree
  - Given, a new data point X
    - Classify X using each of the trees
    - For example, use majority vote: class predicted most often

References

- J. Gehrke, R. Ramakrishnan, and V. Ganti. Rainforest: A framework for fast decision tree construction of large datasets. In Proc. 1998 Int. Conf. Very Large Data Bases, pages 416-427, New York, NY, August 1998.
- M. Kamber, L. Winstone, W. Gong, S. Cheng, and J. Han. Generalization and decision tree induction: Efficient classification in data mining. In Proc. 1997 Int. Workshop Research Issues on Data Engineering (RIDE'97), pages 111-120, Birmingham, England, April 1997.
- M. Mehta, R. Agrawal, and J. Rissanen. SLIQ : A fast scalable classifier for data mining. In Proc. 1996 Int. Conf. Extending Database Technology (EDBT'96), Avignon, France, March 1996.
- J. R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.
- J. R. Quinlan. Bagging, boosting, and c4.5. In Proc. 13th Natl. Conf. on Artificial Intelligence (AAAI'96), 725-730, Portland, OR, Aug. 1996.