2 Nephi 4

16 Behold, my soul delighteth in the things of the Lord; and my heart pondereth continually upon the things which I have seen and heard.

17 Nevertheless, notwithstanding the great goodness of the Lord, in showing me his great and marvelous works, my heart exclaimeth: O wretched man that I am! Yea, my heart sorroweth because of my flesh; my soul grieveth because of mine iniquities.

18 I am encompassed about, because of the temptations and the sins which do so easily beset me.

19 And when I desire to rejoice, my heart groaneth because of my sins; nevertheless, I know in whom I have trusted.

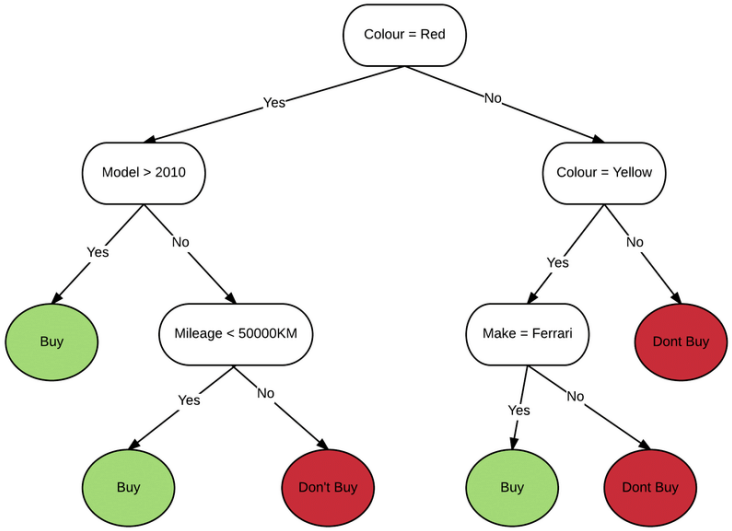
Decision Trees

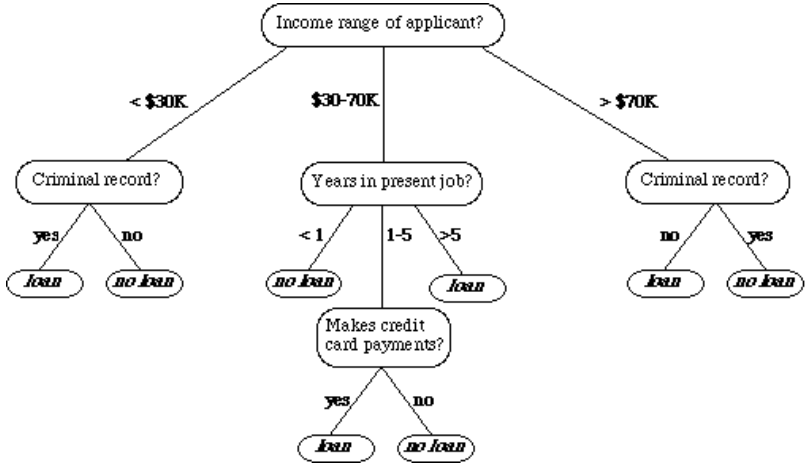
Decision Tree

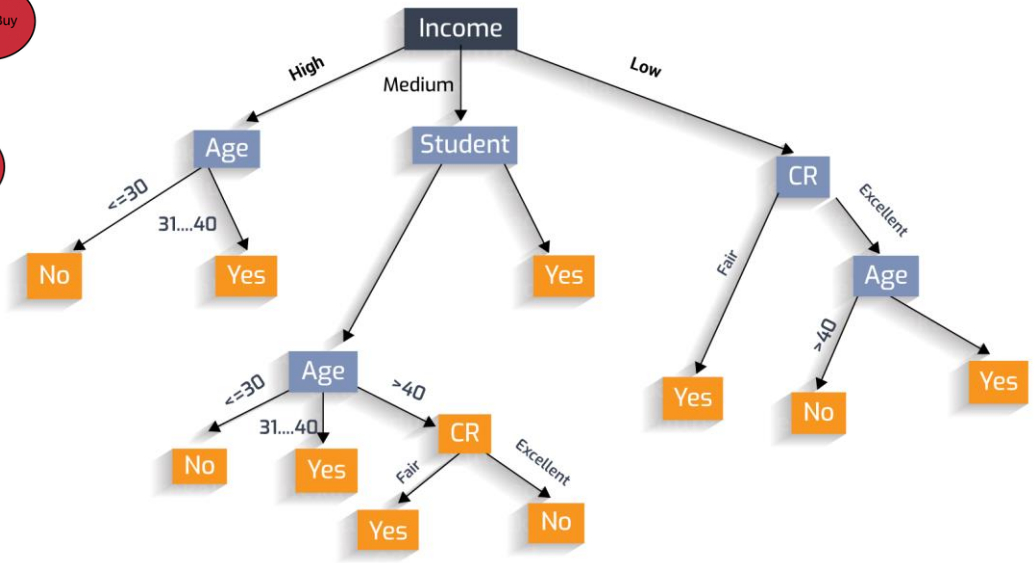
- Decision tree is a classifier in the form of a tree structure
  - Decision node: specifies a test on a single attribute
  - Leaf node: indicates the value of the target attribute
  - Arc/edge: split of one attribute
  - Path: a disjunction of test to make the final decision

- Decision trees classify instances or examples by starting at the root of the tree and moving through it until a leaf node.

Sample Decision Trees



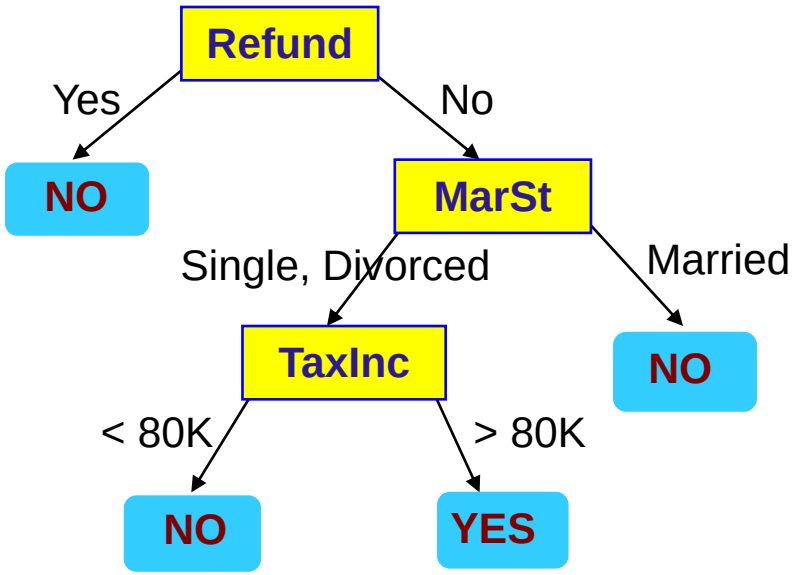




Example of a Decision Tree

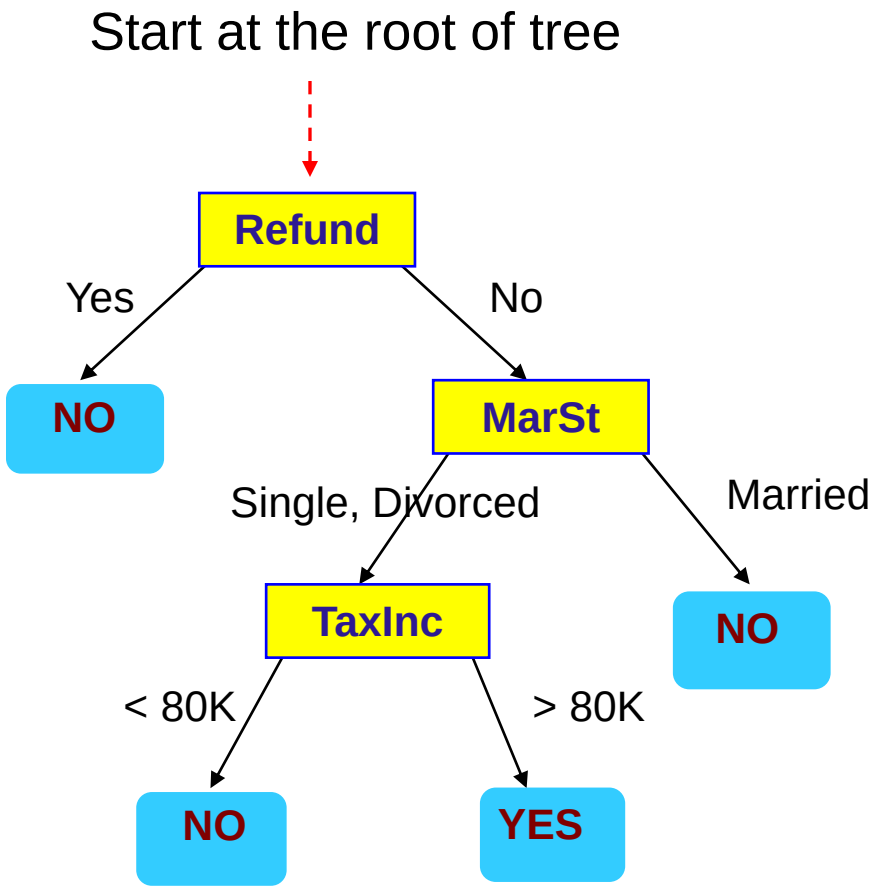
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data



Model: Decision Tree

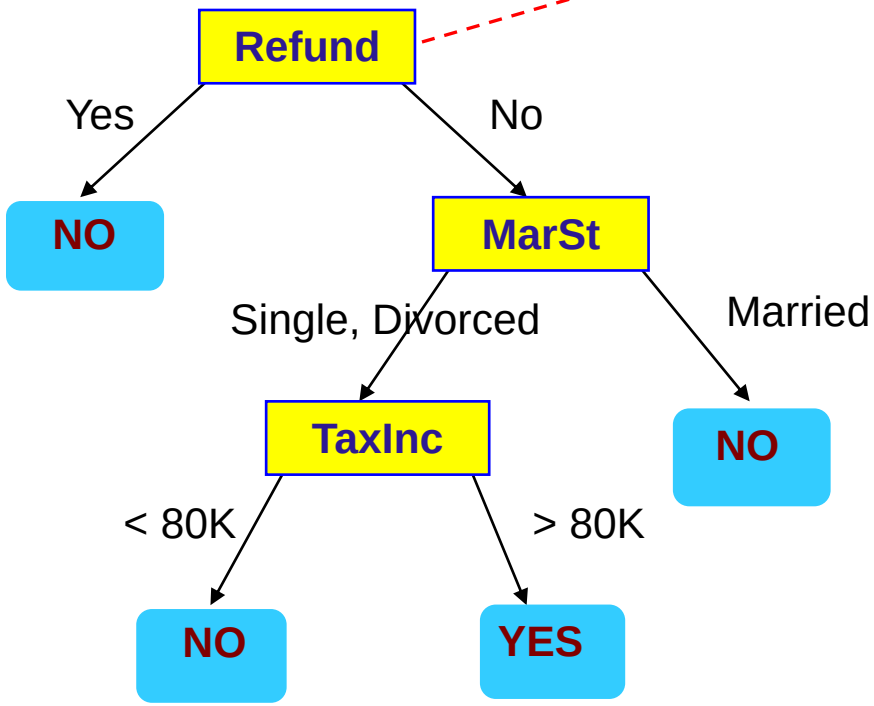
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

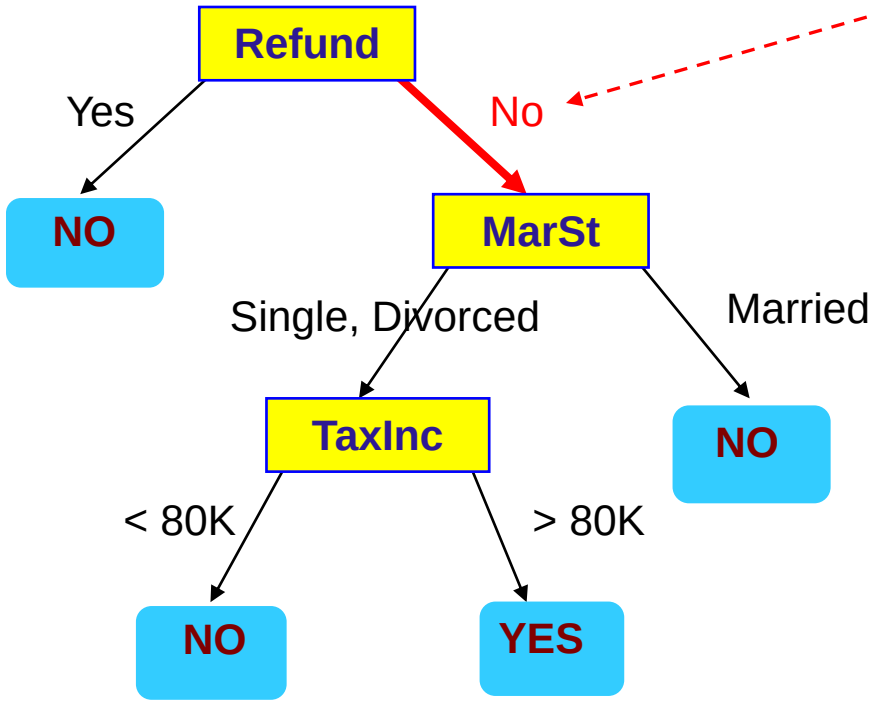
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

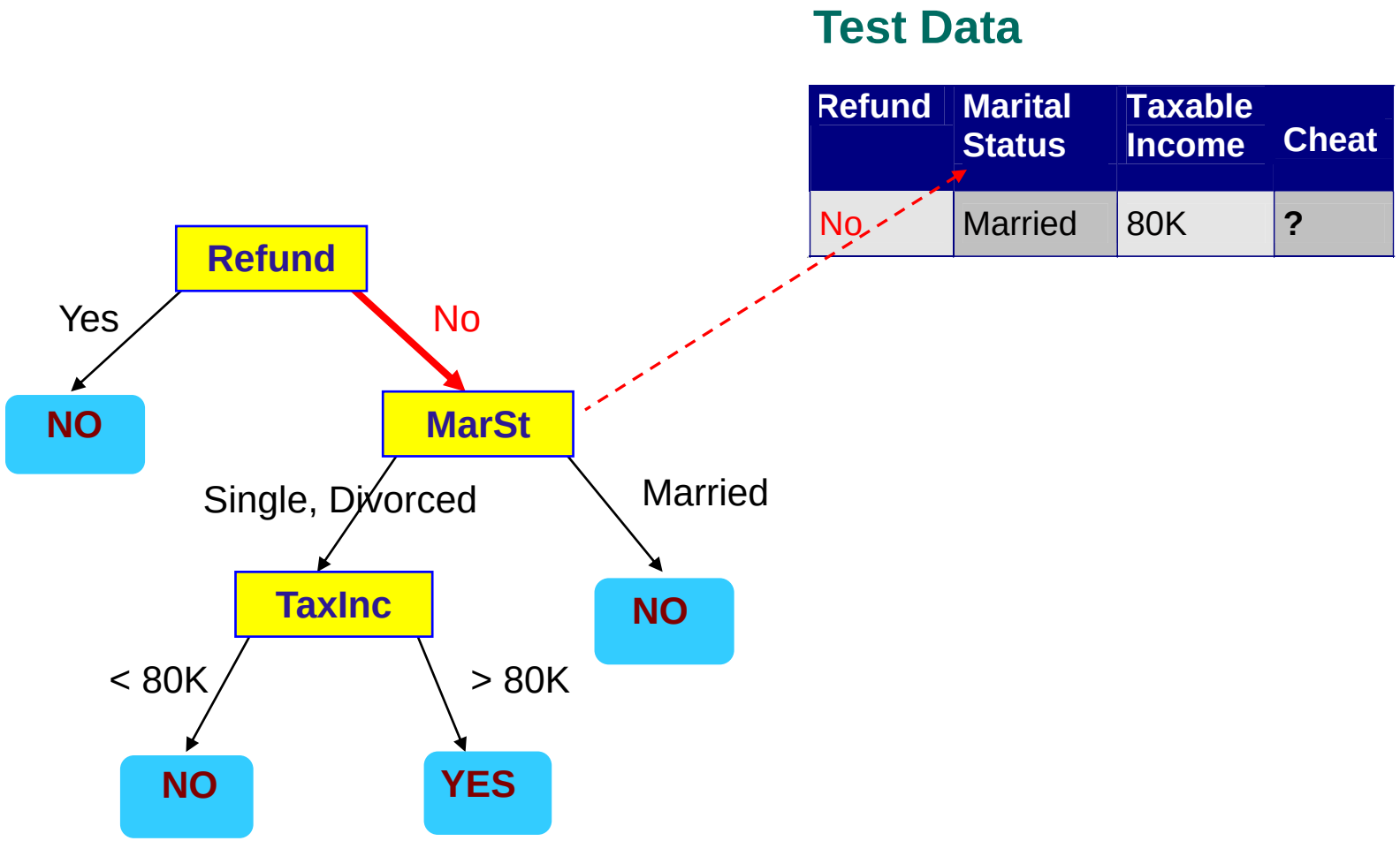
Apply Model to Test Data



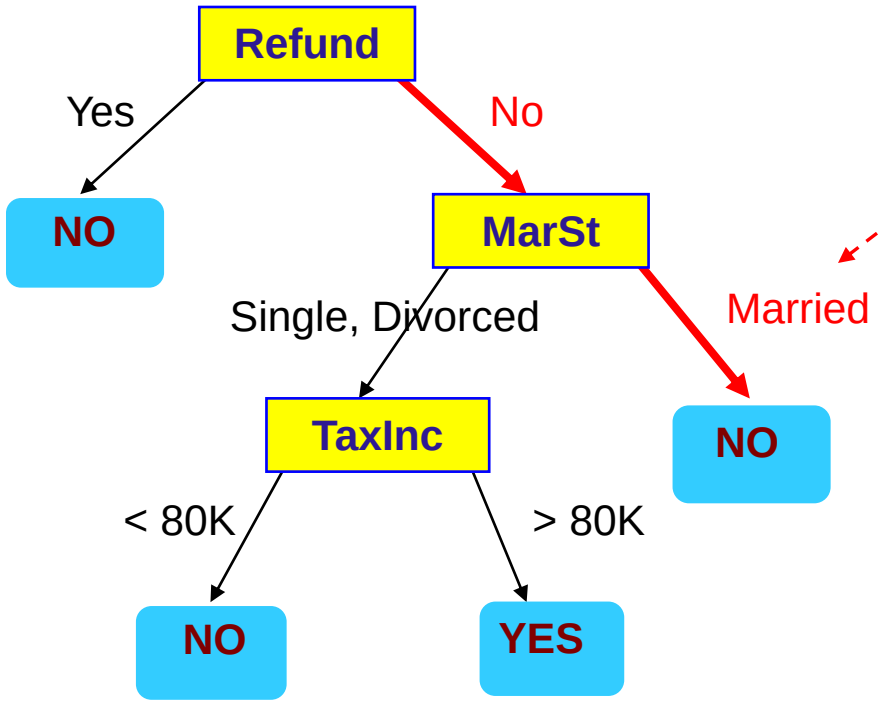
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



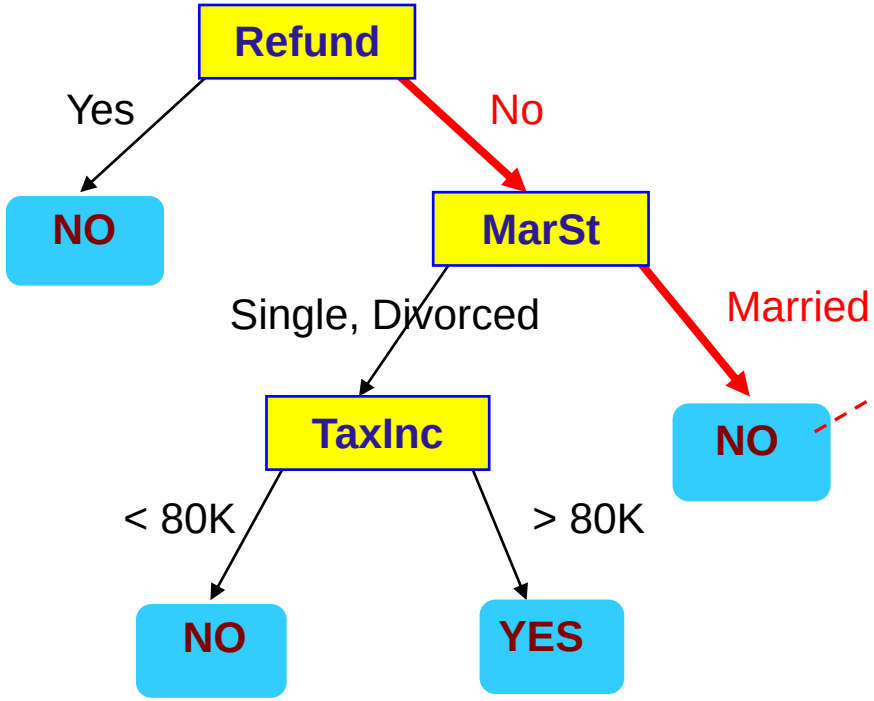
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



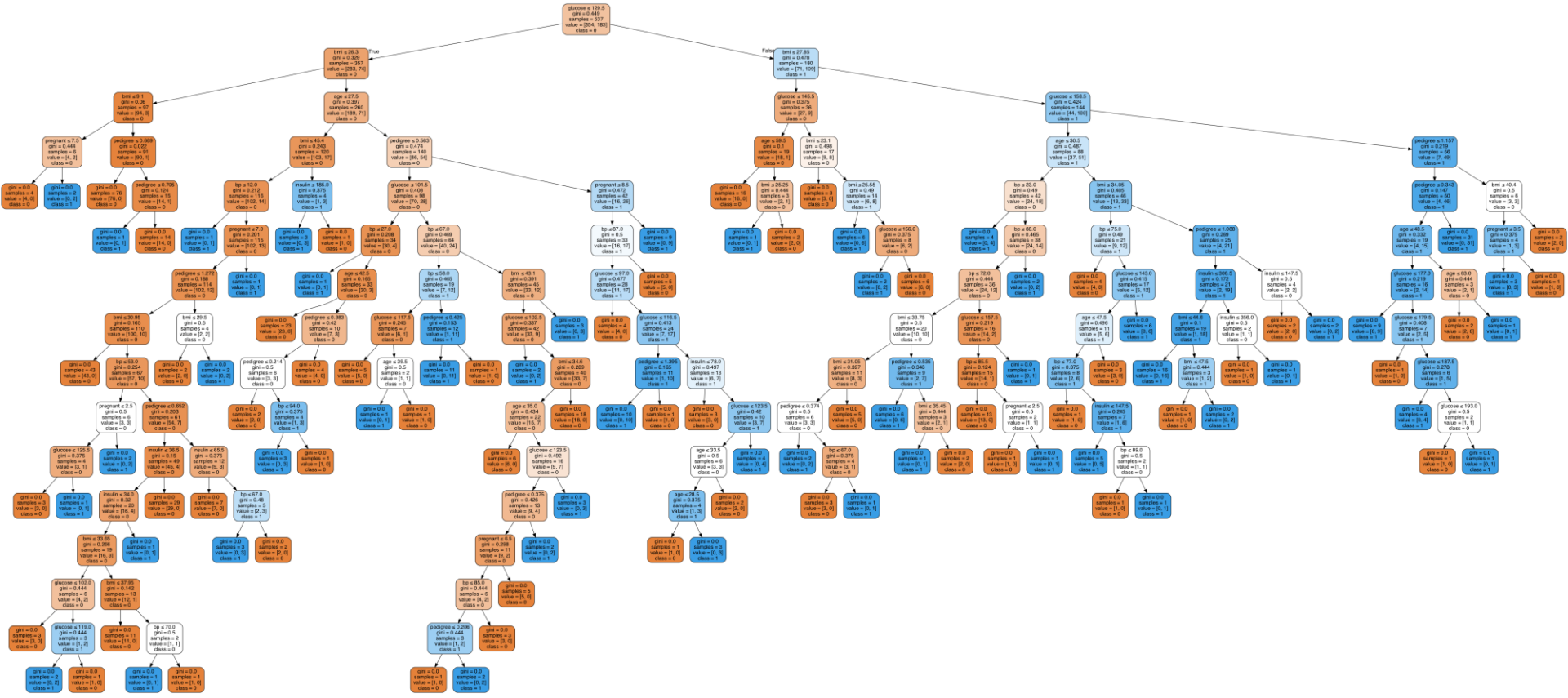
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Assign Cheat to “No”

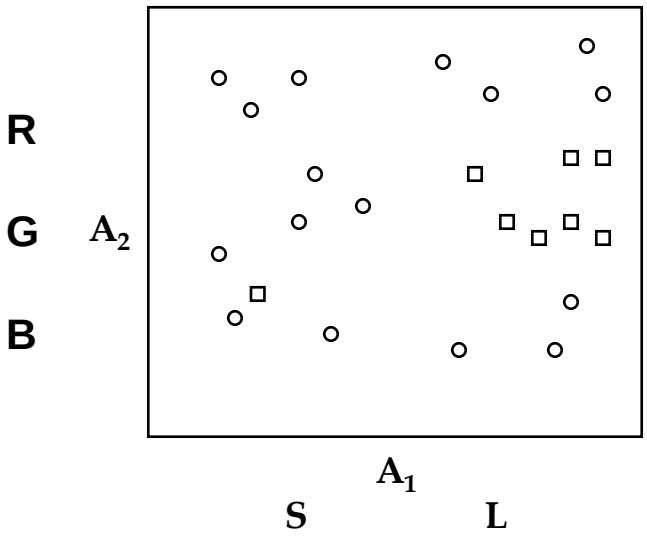
Decision Tree

- Can grow very large and difficult to understand
- Typically they can overfit the training data



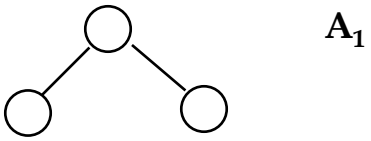
Decision Tree Learning

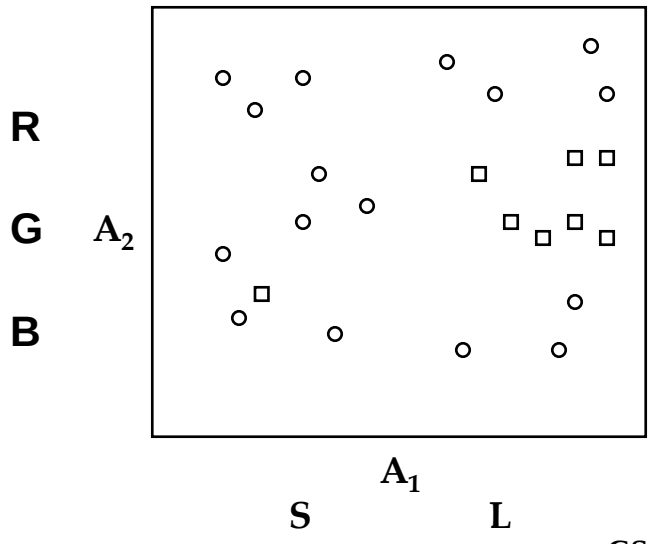
- Assume A_1 is nominal binary feature (Size: S/L)
- Assume A_2 is nominal 3 value feature (Color: R/G/B)
- A goal is to get "pure" leaf nodes. What feature would you split on?

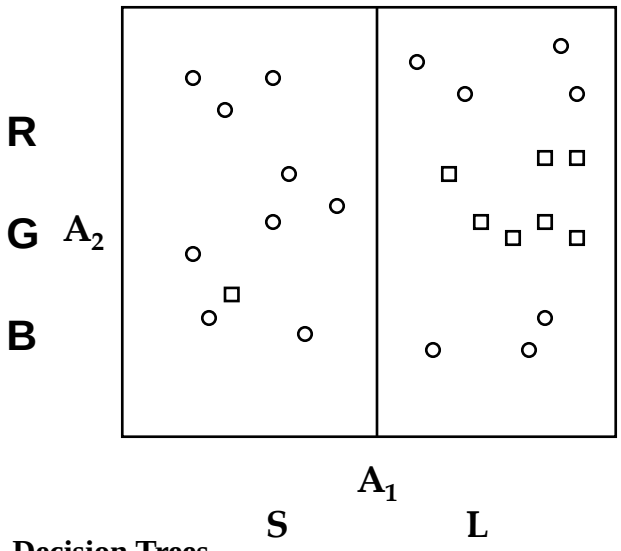


Decision Tree Learning

- Assume A_1 is nominal binary feature (Size: S/L)
- Assume A_2 is nominal 3 value feature (Color: R/G/B)
- Next step for left and right children?

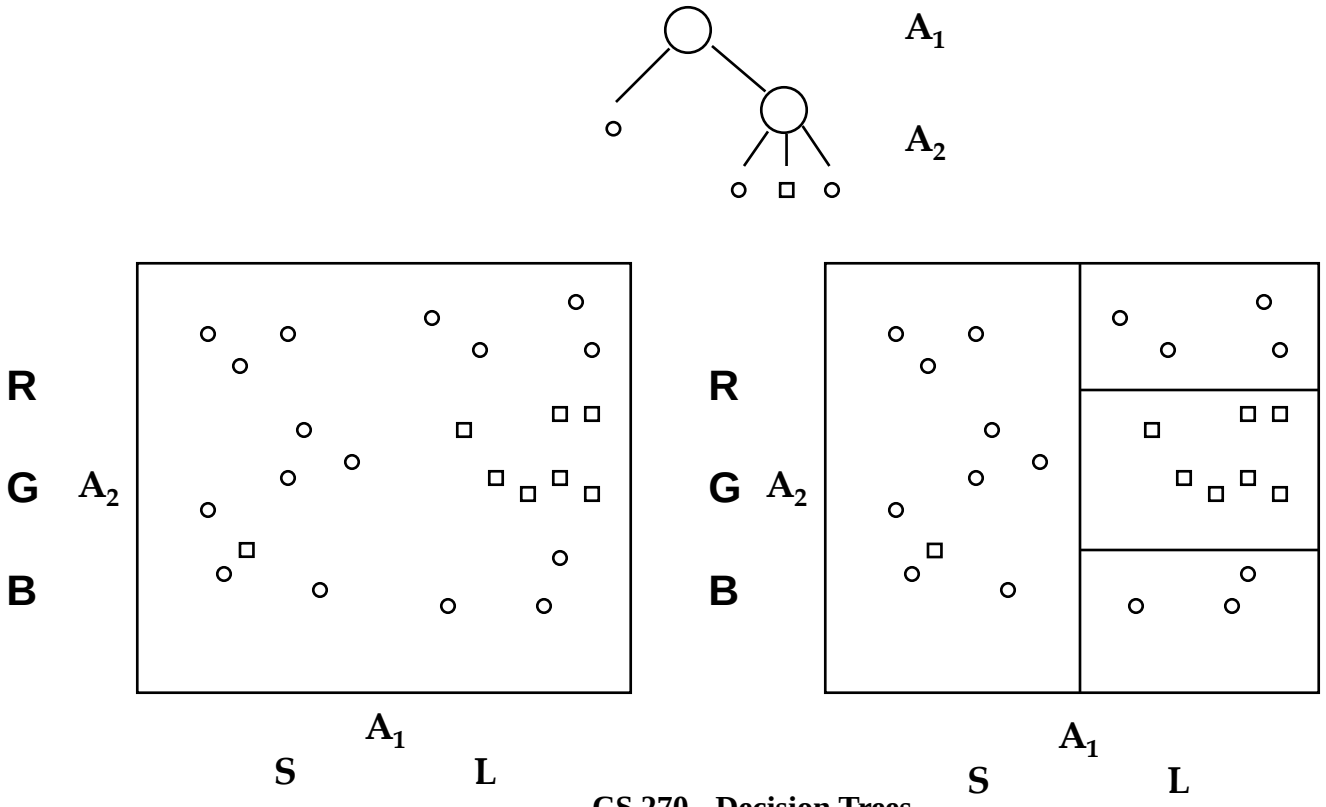






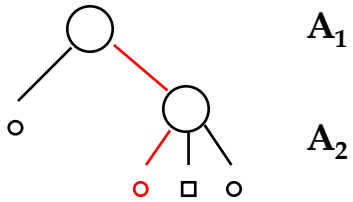
Decision Tree Learning

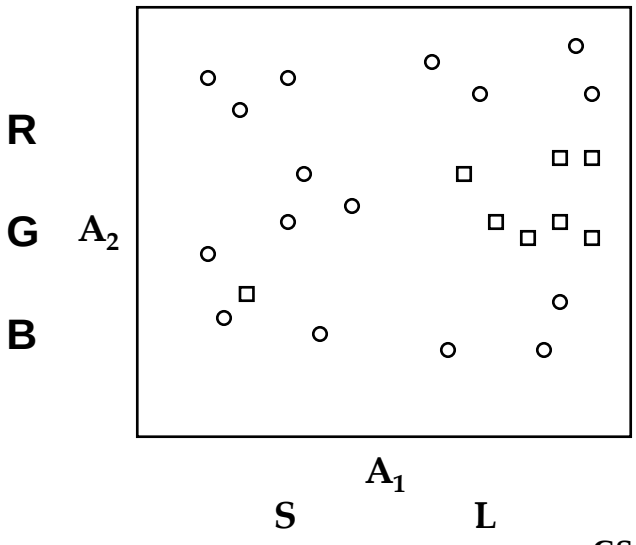
- Assume A_1 is nominal binary feature (Size: S/L)
- Assume A_2 is nominal 3 value feature (Color: R/G/B)
- Decision surfaces are axis aligned Hyper-Rectangles

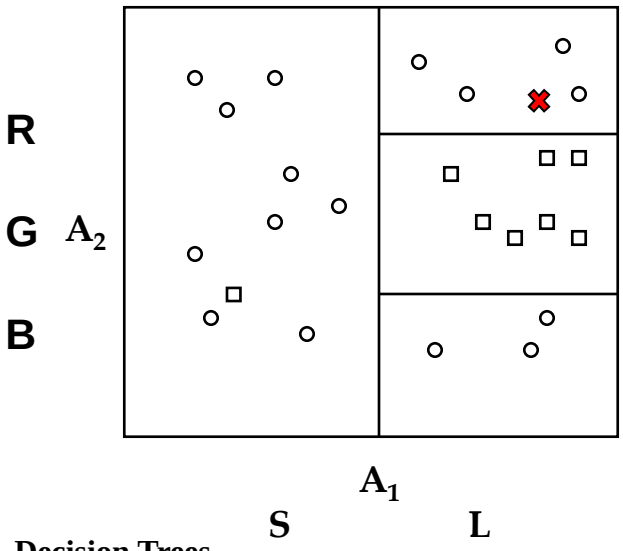


Decision Tree Learning

- Assume A_1 is nominal binary feature (Size: S/L)
- Assume A_2 is nominal 3 value feature (Color: R/G/B)
- Decision surfaces are axis aligned Hyper-Rectangles
- Label leaf nodes with their majority class







Decision Tree Algorithms

- J Ross Quinlan – Australia, ML researcher
  - ID3 (Iterative Dichotimiser 3) – 1986
  - C4.5 – (Version 4.5 written in C) 1993, Handles real valued inputs
  - C5.0 – More efficient implementation
- Leo Breiman - UC Berkeley
  - CART (Classification and Regression Trees) – 1984
    - This is the decision tree approach currently supported in Sklearn
  - Random Forests - 2001
- Independently discovered

Constructing Decision Trees

- **Exponentially many decision trees can be constructed from a given set of attributes**
- **Finding the most accurate tree is NP-hard**
- **In practice: greedy algorithms**
  - **Grow a decision tree by making a series of *locally optimum decisions on which attributes to use* for partitioning the data**

Decision Tree Learning: ID3

- Function ID3(*Training-set*, *Attributes*)
  - If all elements in *Training-set* are in same class, then return leaf node labeled with that class
  - Else if *Attributes* is empty, then return leaf node labeled with majority class in *Training-set*
  - Else if *Training-Set* is empty, then return leaf node labeled with default majority class
  - Else
    - Select and remove *A* from *Attributes*
    - Make *A* the root of the current tree
    - For each value *V* of *A*
      - Create a branch of the current tree labeled by *V*
      - *Partition_V* ← Elements of *Training-set* with value *V* for *A*
      - ID3(*Partition_V*, *Attributes*)
      - Attach result to branch *V*

Decision-tree construction (Example)

	binary	categorical	continuous	class
Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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

Figure 4.6. Training set for predicting borrowers who will default on loan payments.

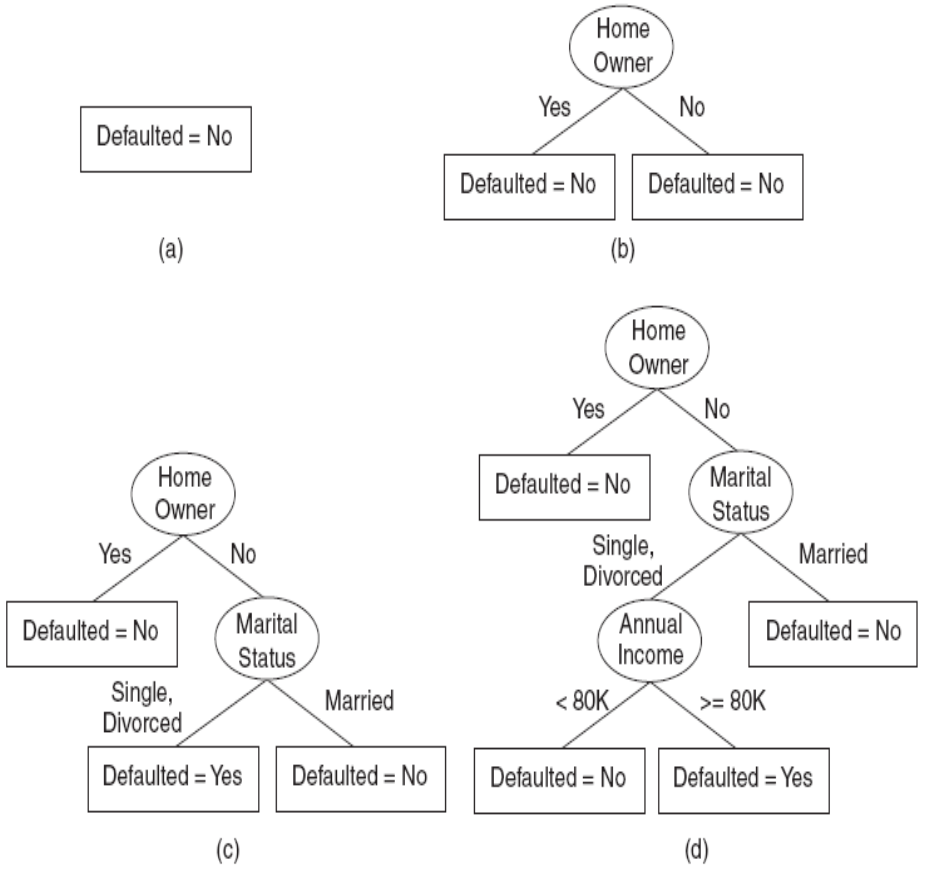


Figure 4.7. Hunt's algorithm for inducing decision trees.

Illustrative Training Set

Risk Assessment for Loan Applications

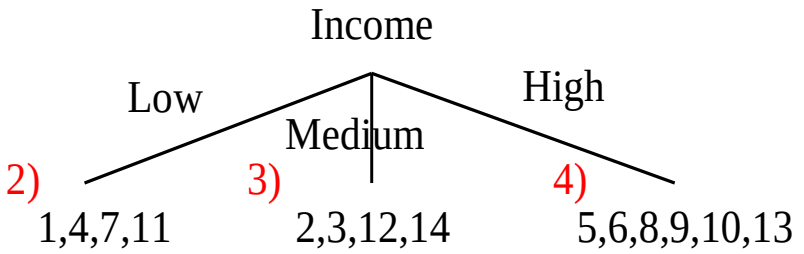
Client #	Credit History	Debt Level	Collateral	Income Level	RISK LEVEL
1	Bad	High	None	Low	**HIGH**
2	Unknown	High	None	Medium	**HIGH**
3	Unknown	Low	None	Medium	**MODERATE**
4	Unknown	Low	None	Low	**HIGH**
5	Unknown	Low	None	High	**LOW**
6	Unknown	Low	Adequate	High	**LOW**
7	Bad	Low	None	Low	**HIGH**
8	Bad	Low	Adequate	High	**MODERATE**
9	Good	Low	None	High	**LOW**
10	Good	High	Adequate	High	**LOW**
11	Good	High	None	Low	**HIGH**
12	Good	High	None	Medium	**MODERATE**
13	Good	High	None	High	**LOW**
14	Bad	High	None	Medium	**HIGH**

ID3 Example

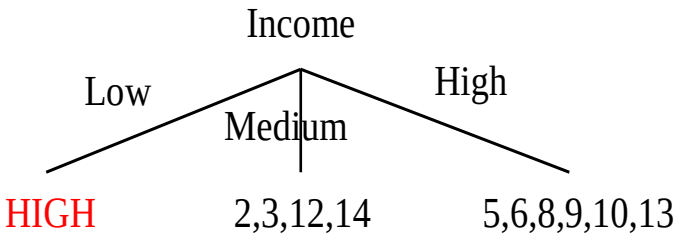
Risk Assessment for Loan Applications

Client #	Credit History	Debt Level	Collateral	Income Level	RISK LEVEL
1	Bad	High	None	Low	**HIGH**
2	Unknown	High	None	Medium	**HIGH**
3	Unknown	Low	None	Medium	**MODERATE**
4	Unknown	Low	None	Low	**HIGH**
5	Unknown	Low	None	High	**LOW**
6	Unknown	Low	Adequate	High	**LOW**
7	Bad	Low	None	Low	**HIGH**
8	Bad	Low	Adequate	High	**MODERATE**
9	Good	Low	None	High	**LOW**
10	Good	High	Adequate	High	**LOW**
11	Good	High	None	Low	**HIGH**
12	Good	High	None	Medium	**MODERATE**
13	Good	High	None	High	**LOW**
14	Bad	High	None	Medium	**HIGH**

1) Choose Income as root of tree.



2) All examples are in the same class, HIGH.
Return Leaf Node.

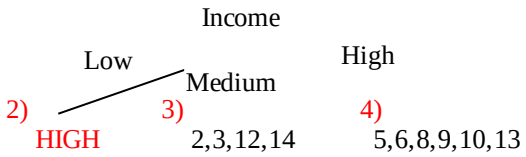


ID3 Example

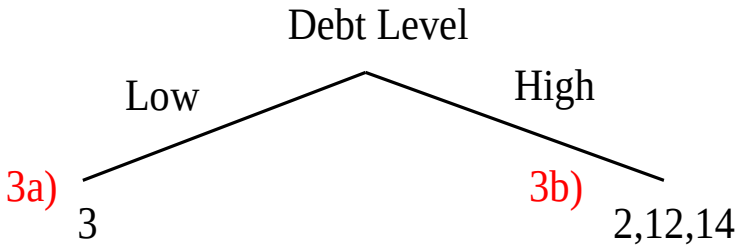
Risk Assessment for Loan Applications

Client #	Credit History	Debt Level	Collateral	Income Level	RISK LEVEL
1	Bad	High	None	Low	**HIGH**
2	Unknown	High	None	Medium	**HIGH**
3	Unknown	Low	None	Medium	**MODERATE**
4	Unknown	Low	None	Low	**HIGH**
5	Unknown	Low	None	High	**LOW**
6	Unknown	Low	Adequate	High	**LOW**
7	Bad	Low	None	Low	**HIGH**
8	Bad	Low	Adequate	High	**MODERATE**
9	Good	Low	None	High	**LOW**
10	Good	High	Adequate	High	**LOW**
11	Good	High	None	Low	**HIGH**
12	Good	High	None	Medium	**MODERATE**
13	Good	High	None	High	**LOW**
14	Bad	High	None	Medium	**HIGH**

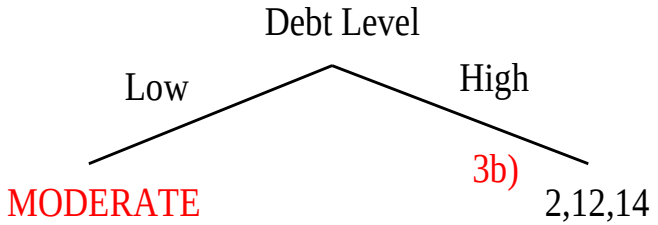
1) Choose Income as root of tree.



3) Choose Debt Level as root of subtree.

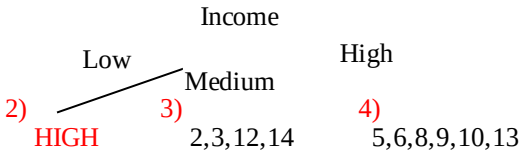


3a) All examples are in the same class, MODERATE.
Return Leaf node.



ID3 Example

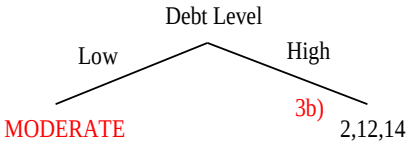
1) Choose Income as root of tree.



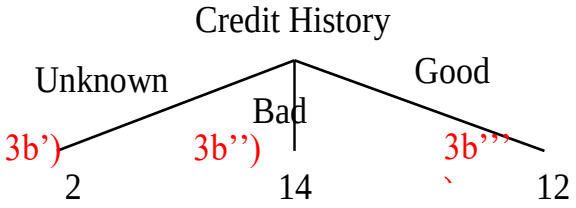
3a) All examples are in the same class, MODERATE.
Return Leaf node.

Risk Assessment for Loan Applications

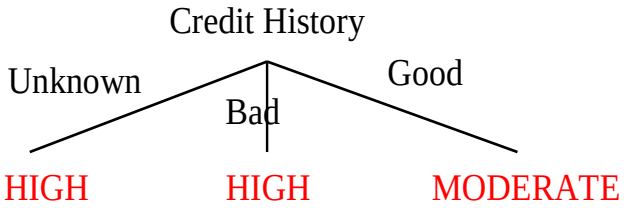
Client #	Credit History	Debt Level	Collateral	Income Level	RISK LEVEL
1	Bad	High	None	Low	**HIGH**
2	Unknown	High	None	Medium	**HIGH**
3	Unknown	Low	None	Medium	**MODERATE**
4	Unknown	Low	None	Low	**HIGH**
5	Unknown	Low	None	High	**LOW**
6	Unknown	Low	Adequate	High	**LOW**
7	Bad	Low	None	Low	**HIGH**
8	Bad	Low	Adequate	High	**MODERATE**
9	Good	Low	None	High	**LOW**
10	Good	High	Adequate	High	**LOW**
11	Good	High	None	Low	**HIGH**
12	Good	High	None	Medium	**MODERATE**
13	Good	High	None	High	**LOW**
14	Bad	High	None	Medium	**HIGH**



3b) Choose Credit History as root of subtree.



3b'-3b''') All examples are in the same class.
Return Leaf nodes.

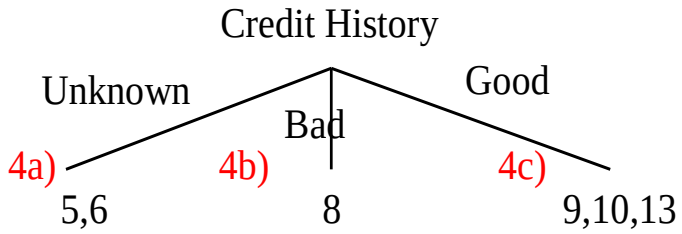


ID3 Example

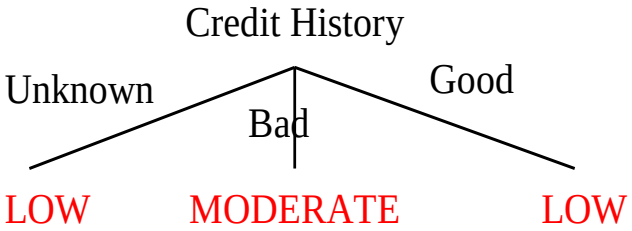
Risk Assessment for Loan Applications

Client #	Credit History	Debt Level	Collateral	Income Level	RISK LEVEL
1	Bad	High	None	Low	**HIGH**
2	Unknown	High	None	Medium	**HIGH**
3	Unknown	Low	None	Medium	**MODERATE**
4	Unknown	Low	None	Low	**HIGH**
5	Unknown	Low	None	High	**LOW**
6	Unknown	Low	Adequate	High	**LOW**
7	Bad	Low	None	Low	**HIGH**
8	Bad	Low	Adequate	High	**MODERATE**
9	Good	Low	None	High	**LOW**
10	Good	High	Adequate	High	**LOW**
11	Good	High	None	Low	**HIGH**
12	Good	High	None	Medium	**MODERATE**
13	Good	High	None	High	**LOW**
14	Bad	High	None	Medium	**HIGH**

4) Choose Credit History as root of subtree.

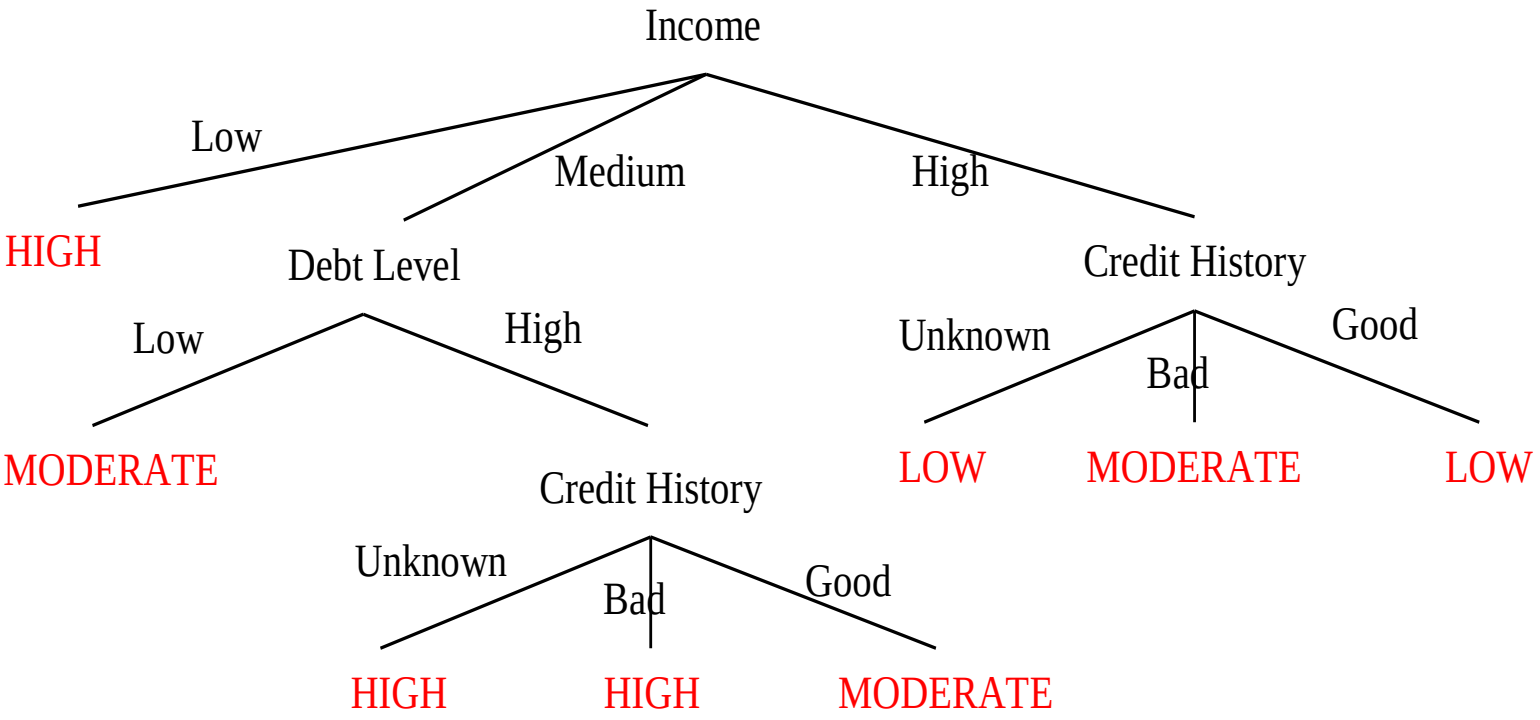


4a-4c) All examples are in the same class.
Return Leaf nodes.



ID3 Example

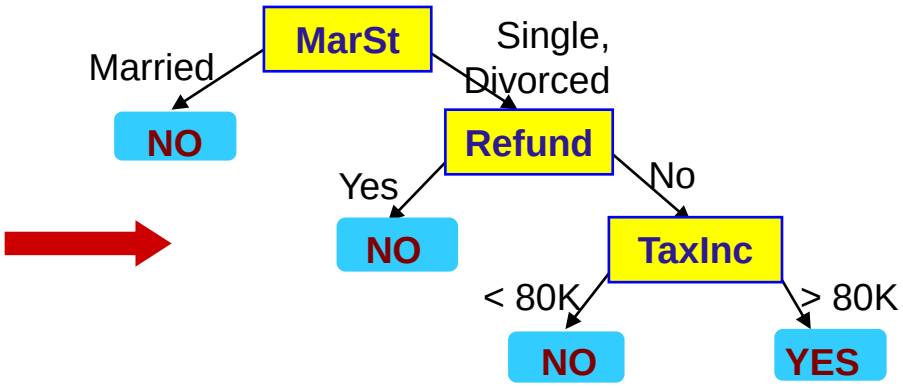
Attach subtrees at appropriate places.



Non-Uniqueness

- Decision trees are not unique:
  - Given a set of training instances *T*, there generally exists a number of decision trees that are consistent with (or fit) *T*

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



ID3's Question

Given a training set, which of all of the decision trees consistent with that training set should we pick?

More precisely:

Given a training set, which of all of the decision trees consistent with that training set has the greatest likelihood of correctly classifying unseen instances of the population?

ID3's (Approximate) Bias

- ID3 (and family) prefers simpler decision trees
- Occam's Razor Principle:
  - "It is vain to do with more what can be done with less...Entities should not be multiplied beyond necessity."
- Intuitively:
  - Always accept the simplest answer that fits the data, avoid unnecessary constraints
  - Simpler trees are more general

ID3's Question Revisited

- Since ID3 builds a decision tree by recursively selecting attributes and splitting the training data based on the values of these attributes

Practically:

Given a training set, how do we select attributes so that the resulting tree is as small as possible, i.e. contains as few attributes as possible?

Not All Attributes Are Created Equal

- Each attribute of an instance may be thought of as contributing a certain amount of information to its classification
  - Think 20-Questions:
    - What are good questions?
    - Ones whose answers maximize information gained
  - For example, determine shape of an object:
    - Num. sides contributes a certain amount of information
    - Color contributes a different amount of information
- ID3 measures information gained by making each attribute the root of the current subtree, and subsequently chooses the attribute that produces the **greatest information gain**

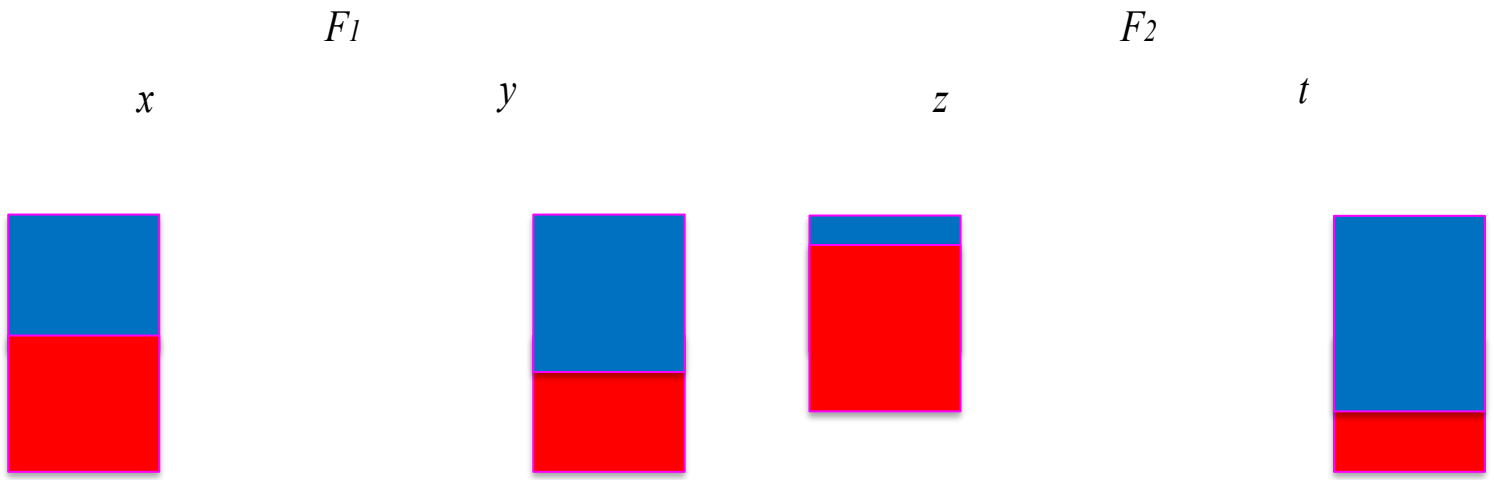
Selecting the best split

- **p(i|t)**: fraction of records associated with node **t** belonging to class **i**
- **Best split** is selected based on the degree of **impurity** of the child nodes
  - Class distribution **(0,1)** has **high purity**
  - Class distribution **(0.5,0.5)** has the **smallest purity (highest impurity)**

- **Intuition:**
  - high purity → small value of impurity measures → better split

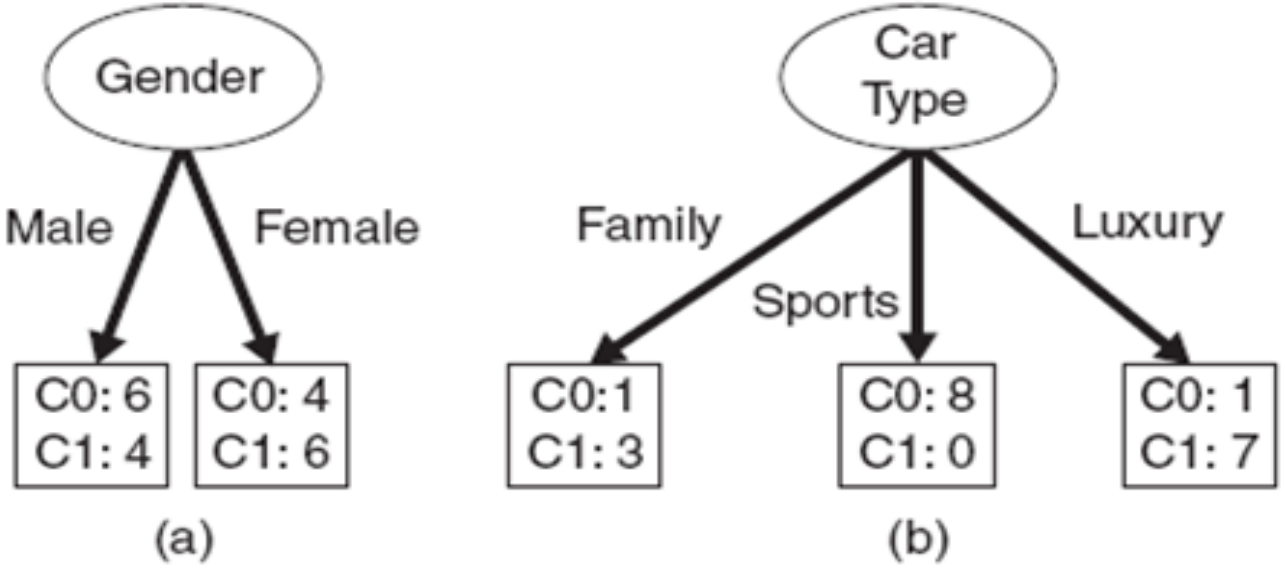
Purity/Information/Entropy (I)

- Assume 2 classes (red/blue) and 2 features F_1 and F_2, each with two values, such that:



- Which of F_1 or F_2 would we choose and why?

Selecting the best split



If you have a choice of these two attributes as the next one to split on, which would you choose?

Impurity measures

- **p(i|t)**: fraction of records associated with node **t** belonging to class **i**

$$\text{Entropy}(t) = -\sum_{i=1}^{c} p(i \mid t) \log p(i \mid t)$$

$$\text{Gini}(t) = 1 - \sum_{i=1}^{c} \left[p(i \mid t) \right]^2$$

$$\text{Classification error}(t) = 1 - \max_i \left[p(i \mid t) \right]$$

Range of impurity measures

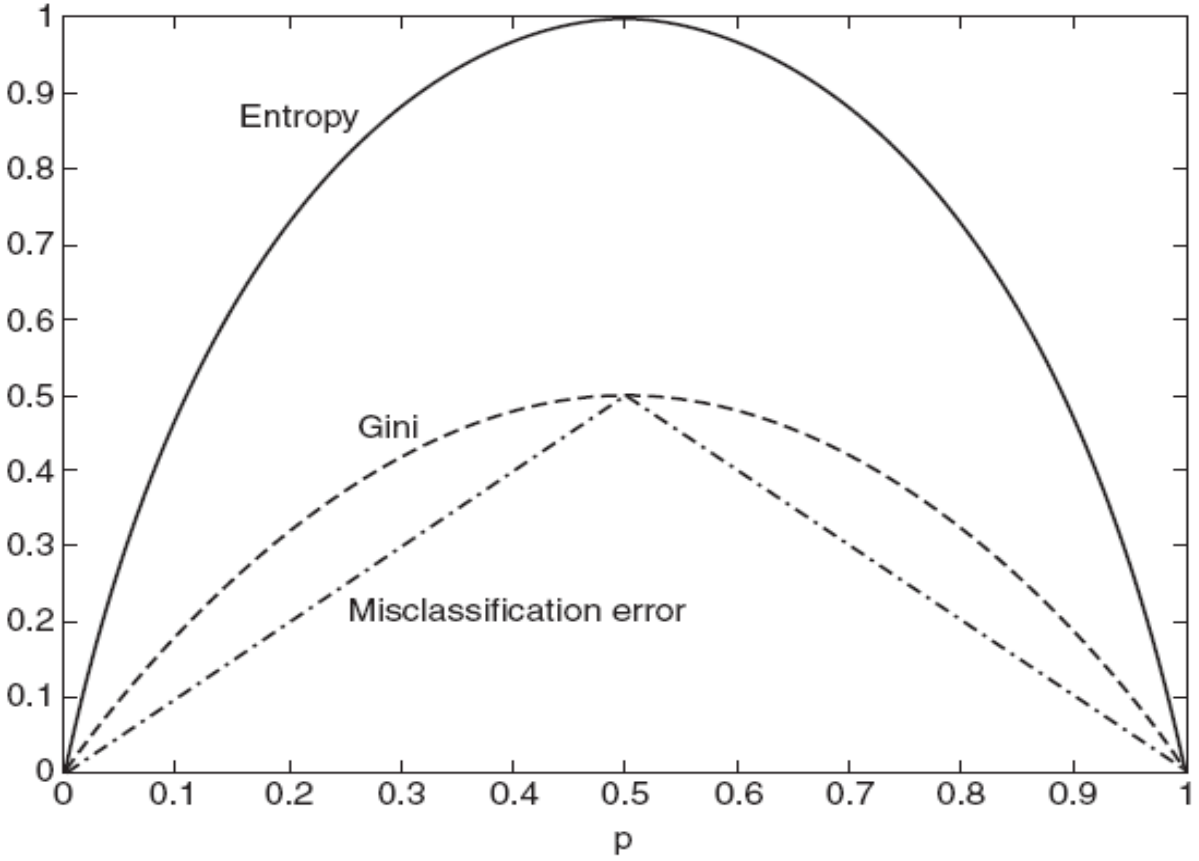


Figure 4.13. Comparison among the impurity measures for binary classification problems.

Impurity measures

- In general the different impurity measures are ***consistent***
- ***Gain of a test condition:*** compare the impurity of the parent node with the impurity of the child nodes

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

- Maximizing the gain == minimizing the weighted average impurity measure of children nodes
- If **I() = Entropy(),** then **Δ_{info}** is called **information gain**

Information Gain

- Information Gain:

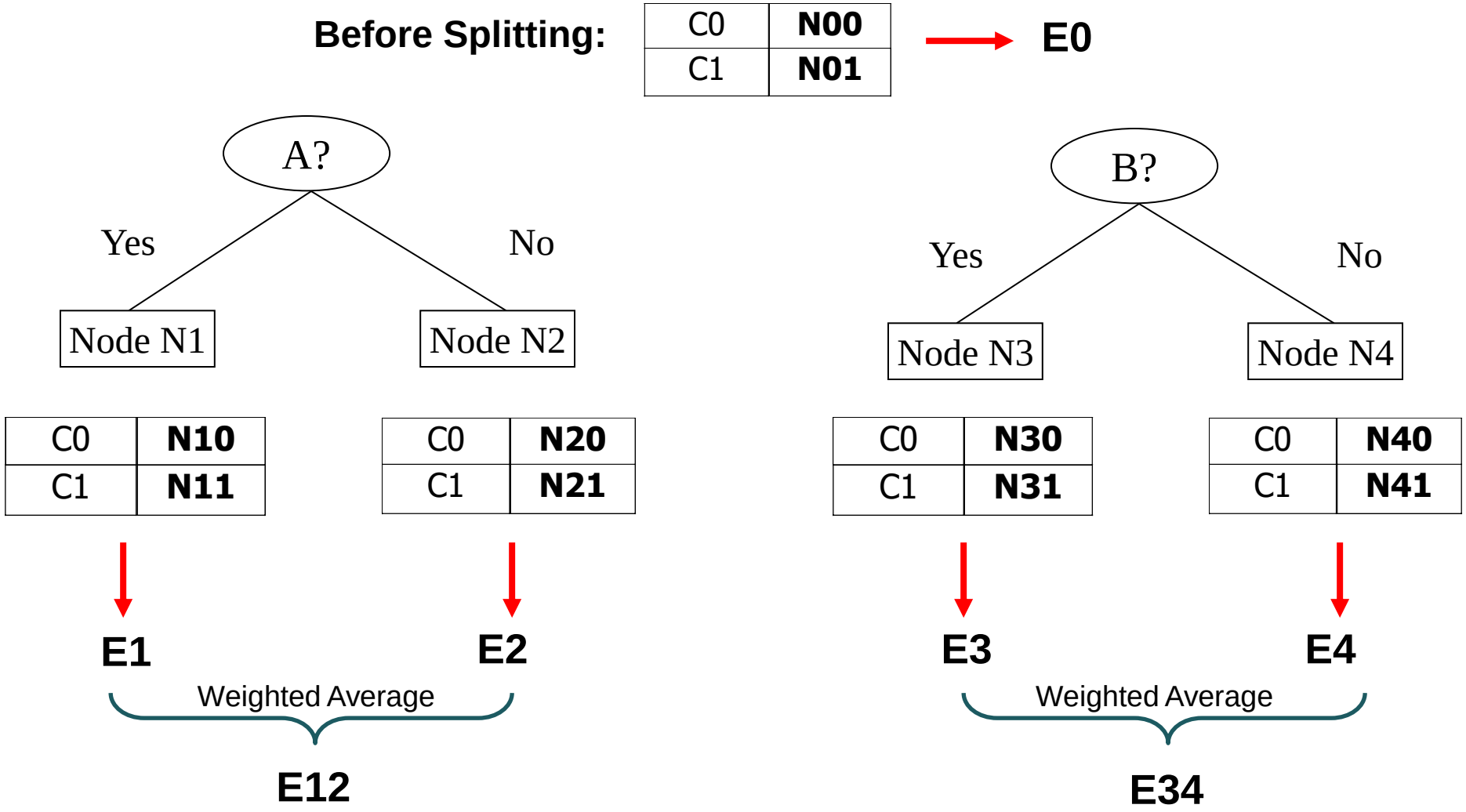
$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

(where parent Node, p is split into k partitions, and

n_i is number of records in partition i)

  - Measures reduction in entropy achieved because of the split ➔ maximize
  - ID3 chooses to split on the attribute that results in the largest reduction, i.e, (maximizes GAIN)
  - Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

Computing Gain



Gain = E0 – E12 vs. E0 – E34

Example

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

1. compute the entropy for data-set
2. for every attribute/feature:
   1. calculate entropy for all categorical values
   2. take average information entropy for the current attribute
   3. calculate gain for the current attribute
3. pick the highest gain attribute.
4. Repeat until we get the tree we desired.

$$H(S) = \sum_{c \in C} -p(c) \log_2 p(c)$$

C ={yes,no}

Out of 14 instances, 9 are classified as yes, and 5 as no

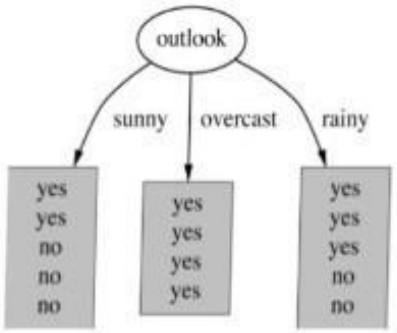
pyes = −(9/14)*log2(9/14) = 0.41

pno = −(5/14)*log2(5/14) = 0.53

H (S) = pyes + pno = 0.94

Example

For every feature, calculate the entropy and the information gain



E (Outlook=sunny) = $-\frac{2}{5}\log\left(\frac{2}{5}\right)-\frac{3}{5}\log\left(\frac{3}{5}\right)=0.971$

E (Outlook=overcast) = $-1\log(1)-0\log(0)=0$

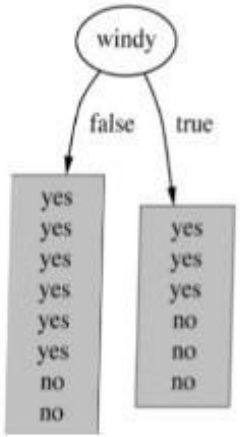
E (Outlook=rainy) = $-\frac{3}{5}\log\left(\frac{3}{5}\right)-\frac{2}{5}\log\left(\frac{2}{5}\right)=0.971$

} H(S,Outlook)

Average Entropy information for Outlook

I (Outlook) = $\frac{5}{14}*0.971+\frac{4}{14}*0+\frac{5}{14}*0.971=0.693$ } $\sum_{t\in T} p(t)H(t)$

Gain (Outlook) = E(S) – I (outlook) = 0.94-.693 = 0.247 ⇨ $IG(A,S)=H(S)-\sum_{t\in T}p(t)H(t)$



E (Windy=false) = $-\frac{6}{8}\log\left(\frac{6}{8}\right)-\frac{2}{8}\log\left(\frac{2}{8}\right)=0.811$

E (Windy=true) = $-\frac{3}{6}\log\left(\frac{3}{6}\right)-\frac{3}{6}\log\left(\frac{3}{6}\right)=1$

Average entropy information for Windy

I (Windy) = $\frac{8}{14}*0.811+\frac{6}{14}*1=0.892$

Gain (Windy) = E(S) – *I* (Windy) = 0.94-0.892=0.048

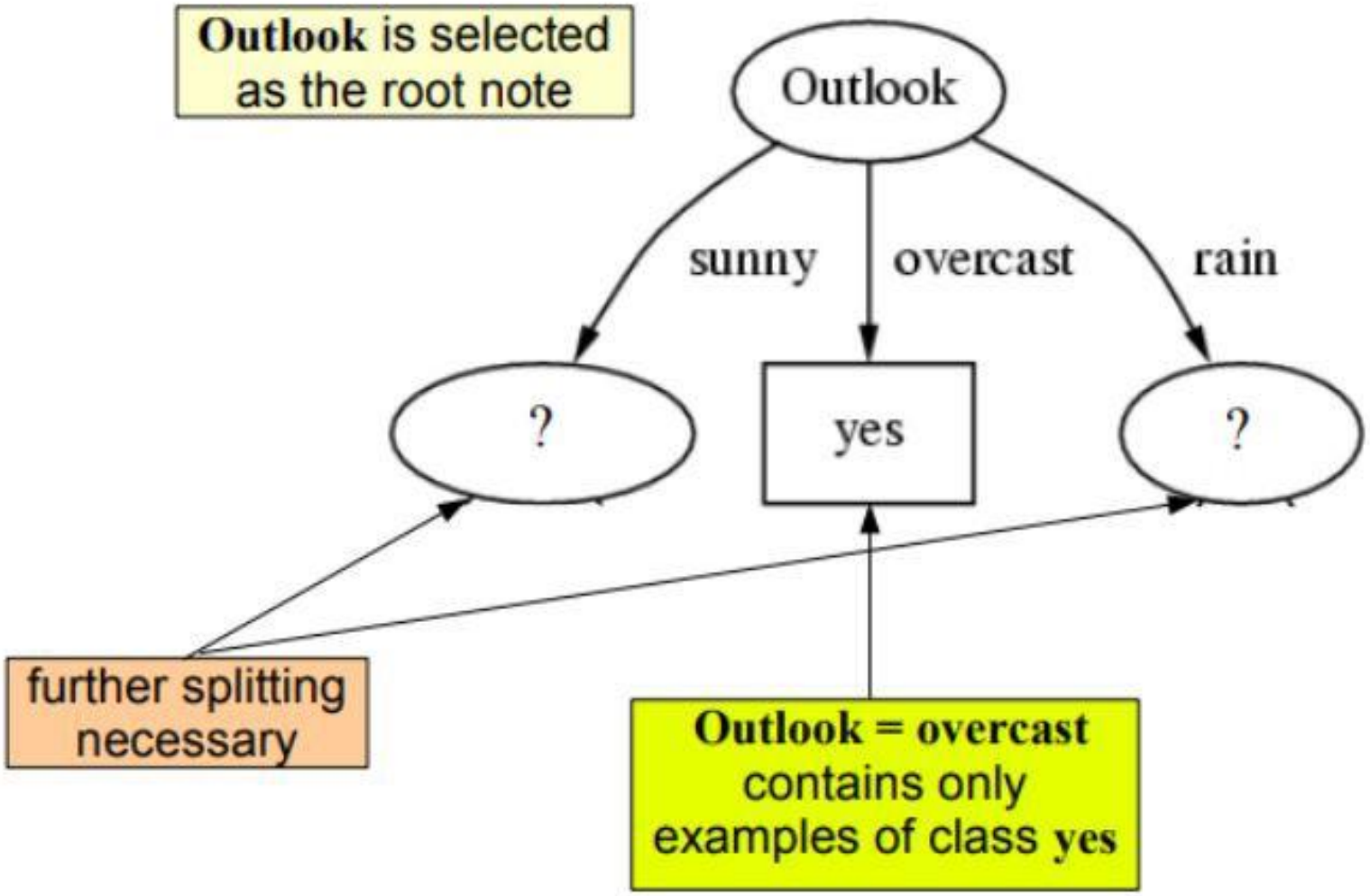
Similarly, calculate the entropy and the information gain for Humidity and Temperature

Example

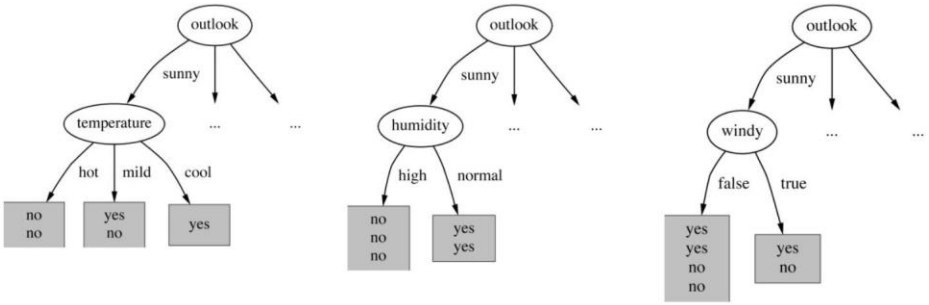
Pick the highest Gain attribute

Outlook		Temperature	
Info:	0.693	Info:	0.911
Gain: 0.940-0.693	0.247	Gain: 0.940-0.911	0.029
Humidity		**Windy**	
Info:	0.788	Info:	0.892
Gain: 0.940-0.788	0.152	Gain: 0.940-0.892	0.048

Example

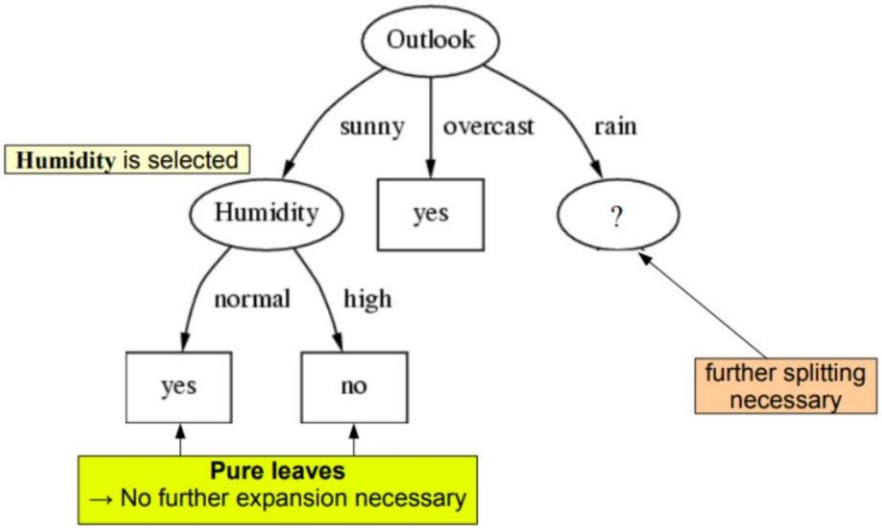

Outlook is selected as the root note
Outlook
sunny
overcast
rain
?
yes
?
further splitting necessary
Outlook = overcast contains only examples of class yes

Example



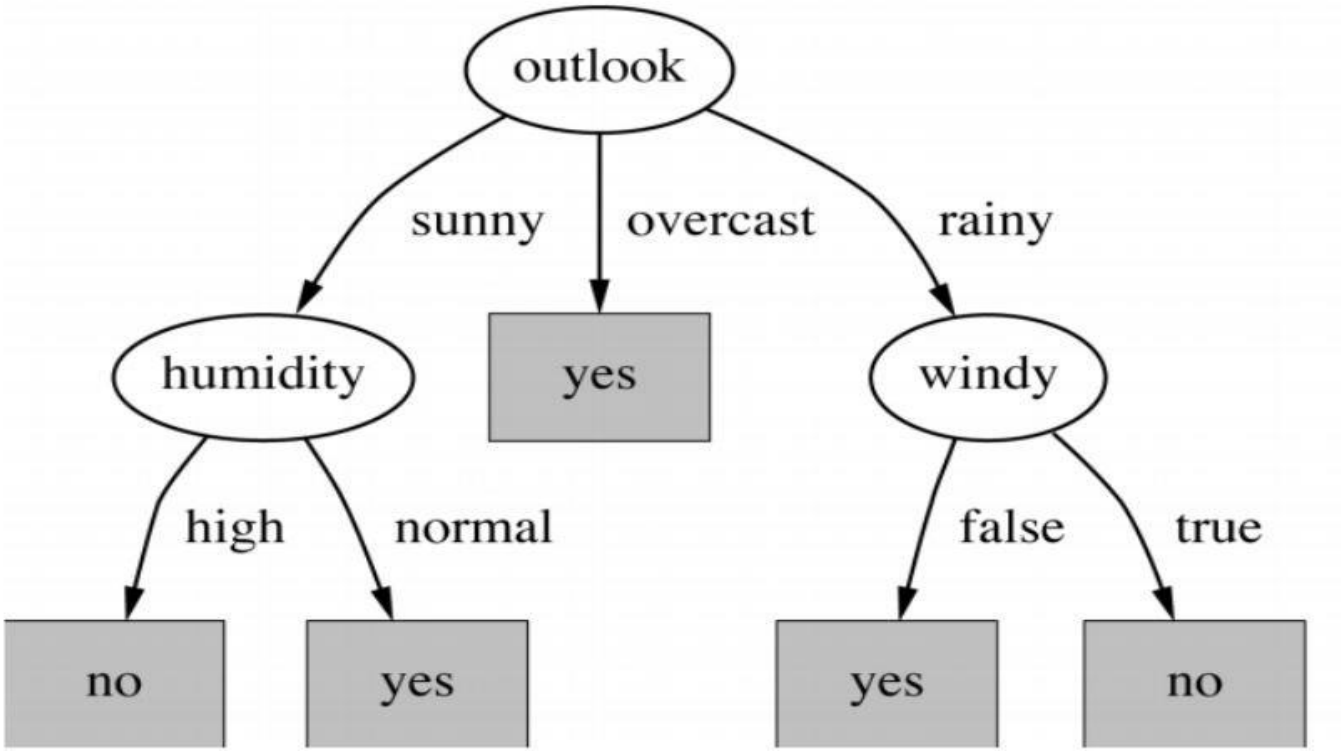
Gain(*Temperature*) = 0.571 bits
Gain(*Humidity*) = 0.971 bits
Gain(*Windy*) = 0.020 bits

Humidity is selected



Example

Final decision tree



Issue with Information Gain

- If F has random values (e.g., SS#), and ends up with only 1 example in each partition, the corresponding split would have maximum information gain, yet be most useless
- Preference for simpler trees would suggest seeking trees with fewer overall nodes, hence features with fewer possible values might be given some kind of preference
  - Use only binary features, but leads to deeper trees, and exponential growth in possible ways of splitting feature sets
  - Use some kind of penalty for features with many values
  - Use a different splitting criterion

Gain Ratio

- Gain Ratio:

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO} \qquad SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

(where parent Node, p is split into k partitions, and

n_i is number of records in partition i)

  - Designed to overcome the disadvantage of GAIN
  - Adjusts GAIN by the entropy of the partitioning (SplitINFO)
  - Higher entropy partitioning (large number of small partitions) is penalized
  - Used by C4.5 (an extension of ID3)

Other Splitting Criterion: GINI Index

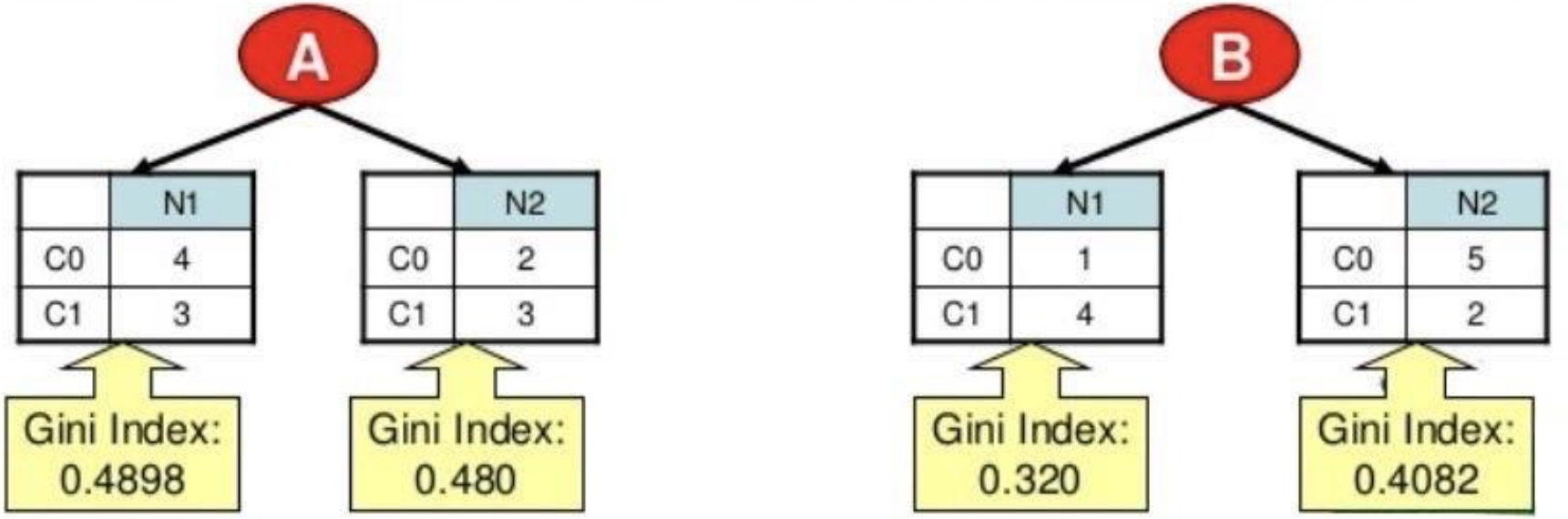
- GINI Index for a given node t :

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

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

  - 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
  - Minimize GINI split value
  - Used by CART, SLIQ, SPRINT

Suppose there are two ways (A and B) to split the data into smaller subset.



$$1-\left[\left(\frac{4}{7}\right)^2+\left(\frac{3}{7}\right)^2\right]$$ **0.4898**

$$1-\left[\left(\frac{2}{5}\right)^2+\left(\frac{3}{5}\right)^2\right]$$ **0.48**

$$1-\left[\left(\frac{1}{5}\right)^2+\left(\frac{4}{5}\right)^2\right]$$ **0.32**

$$1-\left[\left(\frac{5}{7}\right)^2+\left(\frac{2}{7}\right)^2\right]$$ **0.4082**

Now compute the weighted average for each proposed split

$$\frac{7}{12}*0.4898+\frac{5}{12}*0.48$$ **0.4857**

$$\frac{5}{12}*0.32+\frac{7}{12}*0.4082$$ **0.37145**

Choose the minimum

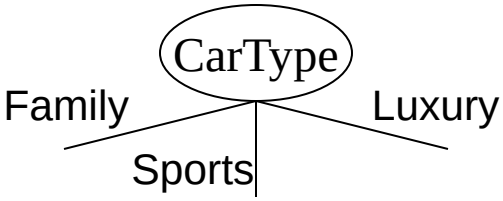
How to Specify Test Condition?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous

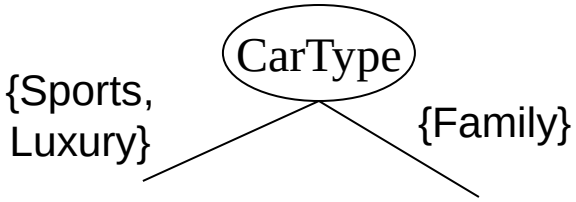
- Depends on number of ways to split
  - Binary split
  - Multi-way split

Splitting Based on Nominal Attributes

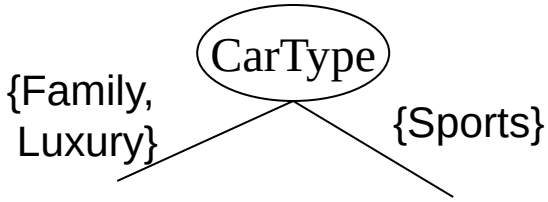
- Multi-way split: Use as many partitions as values



- Binary split: Divide values into two subsets



OR



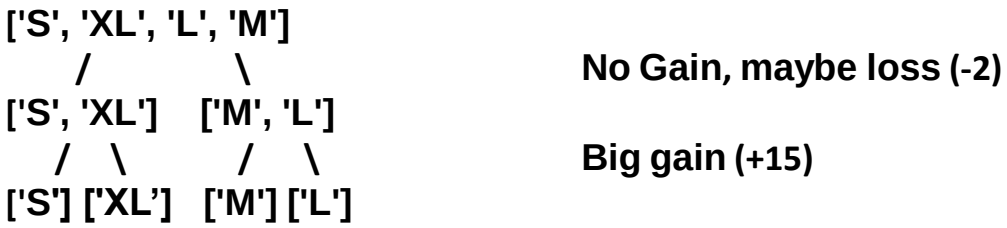
Need to find optimal partitioning!

For Ordinal attributes, make sure the splitting doesn't violate the order.
Or, does order matter?

Splitting Based on Ordinal Values

- Ordinal values imply ordering
- Can just split like nominal but force ordering
- Or, does ordering matter?

Consider predicting weight based on T-shirt size



If we don't consider ordering, the tree must become deeper

We could be fairly accurate with just one level if we considered ordering

Splitting Based on Continuous Attributes

- Different ways of handling
  - Multi-way split: form ordinal categorical attribute
    - Static – discretize once at the beginning
    - Dynamic – repeat on each new partition
  - Binary split: $(A < v)$ or $(A \geq v)$
    - How to choose v?

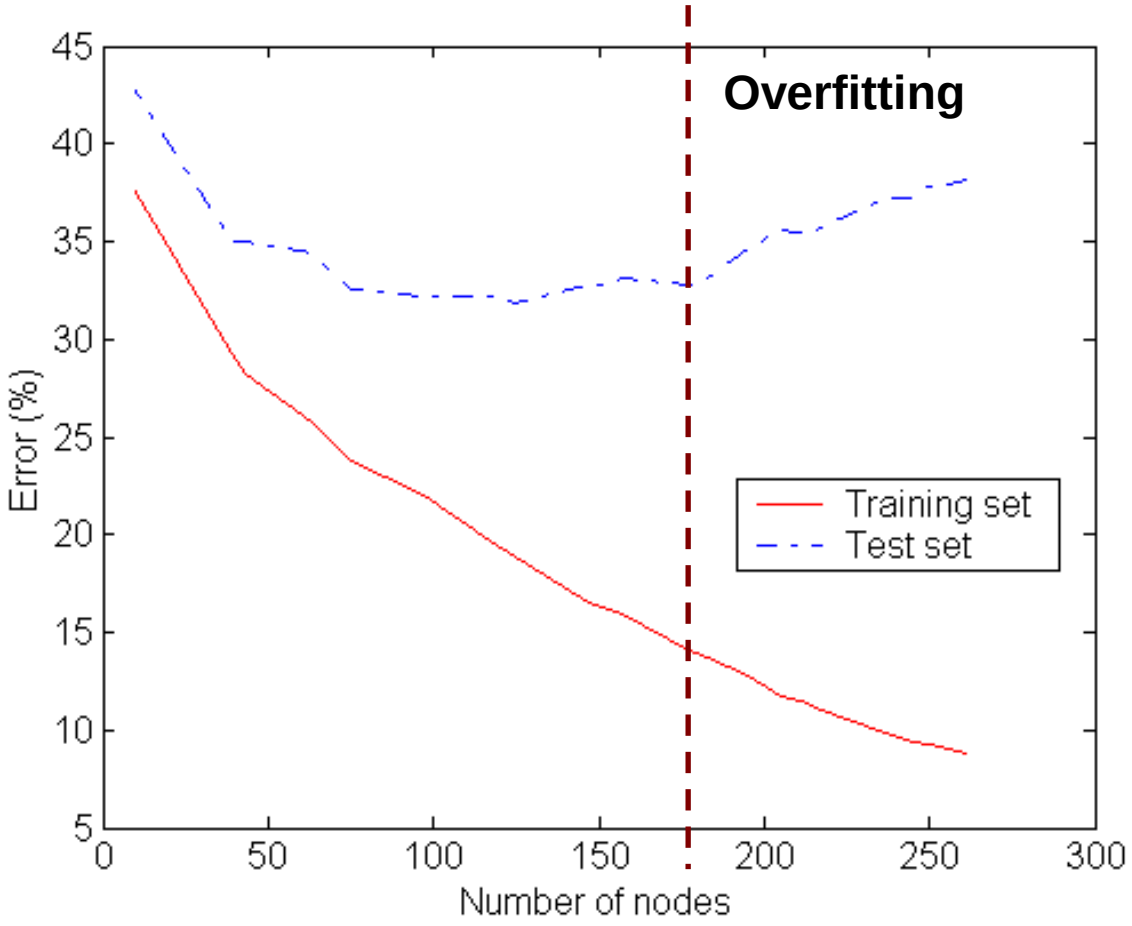
Need to find optimal partitioning!

Can use GAIN or GINI !

Overfitting and Underfitting

- Overfitting:
  - Given a model space H, a specific model $h \in H$ is said to overfit the training data if there exists some alternative model $h' \in H$, such that h has smaller error than h' over the training examples, but h' has smaller error than h over the entire distribution of instances
- Underfitting:
  - The model is too simple, so that both training and test errors are large

Detecting Overfitting



Overfitting in Decision Tree Learning

- Overfitting results in decision trees that are more complex than necessary
  - Tree growth went too far
  - Number of instances gets smaller as we build the tree (e.g., several leaves match a single example)

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

Avoiding Tree Overfitting – Solution 1

- 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 χ^2 test)
    - Stop if expanding the current node does not improve impurity measures (e.g., GINI or GAIN)

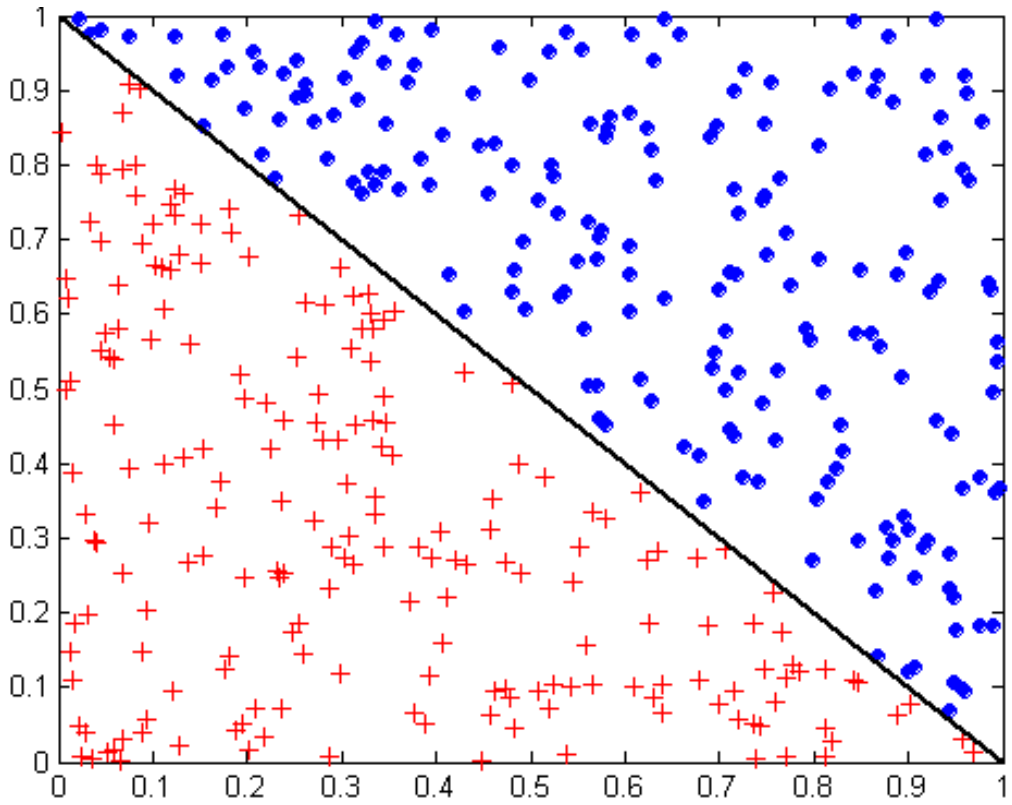
Avoiding Tree Overfitting – Solution 2

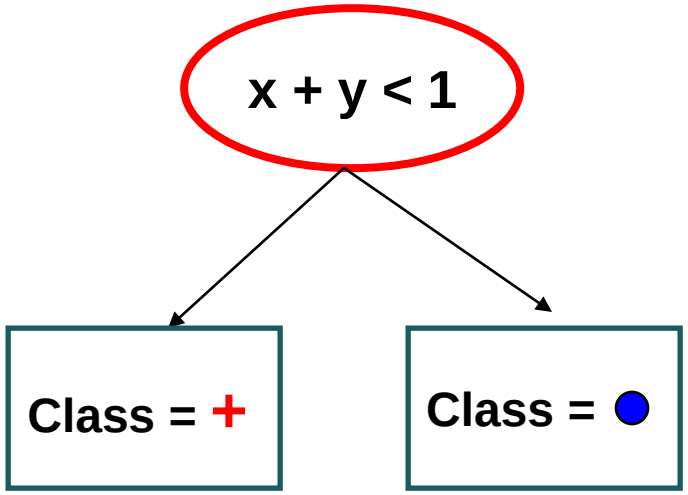
- Post-pruning
  - Split dataset into training and validation sets
  - Grow full decision tree on training set
  - While the accuracy on the validation set increases:
    - Evaluate the impact of pruning each subtree, replacing its root by a leaf labeled with the majority class for that subtree
    - Replace subtree that most increases validation set accuracy (greedy approach)

Decision Tree Based Classification

- Advantages:
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Good accuracy
- Disadvantages:
  - Axis-parallel decision boundaries
  - Redundancy
  - **Need data to fit in memory**
  - **Need to retrain with new data**

Oblique Decision Trees





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

Decision Trees and Big Data

- For each node in the decision tree we need to scan the entire data set
- Could just do multiple passes across the entire data set
  - Ouch!
- Have each processor build a decision tree based on local data
  - Combine decision trees, How?
  - Better?

Combining Decision Trees

- Meta Decision Tree
  - Somehow decide which decision tree to use
  - Hierarchical
  - More training needed
- Voting
  - Run classification data through all decision trees and then vote
- Merge Decision Trees