Machine Learning

BS/MS (Computer Science)

IQRA UNIVERSITY IU

Lecture-07-08
Summer-2014

Classification Using Decision Trees

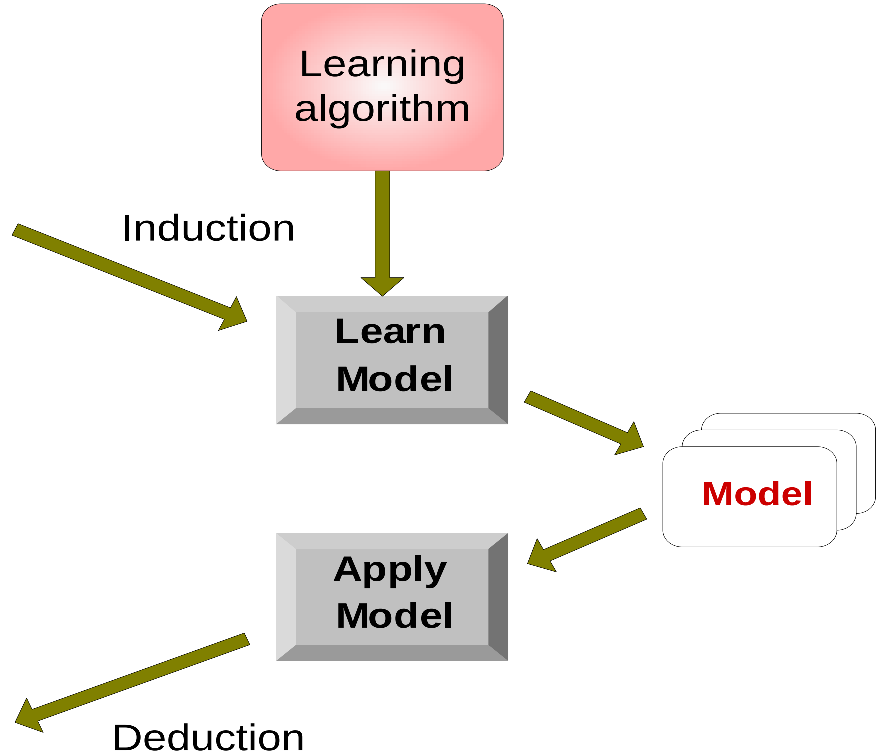
Classification: Definition

- Given a collection of records (*training set*)
  - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for class attribute as a function of the values of other attributes.
- Goal: previously unseen records should be assigned a class as accurately as possible.
  - A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

Illustrating Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Classification Techniques

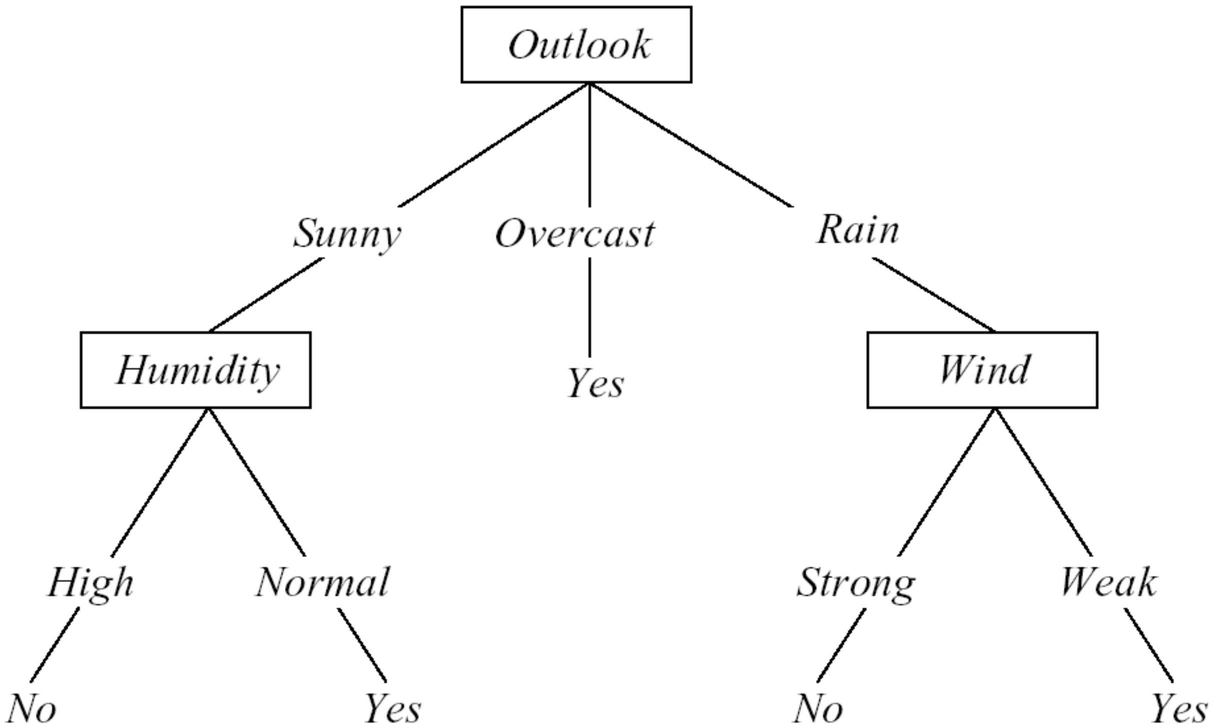
- Decision Tree based Methods
- Rule-based Methods
- Memory based reasoning
- Neural Networks
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines

Decision Tree Learning

Decision Tree Learning

- Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree.
- Learned trees can also be re-represented as set of IF-THEN rules to improve human readability

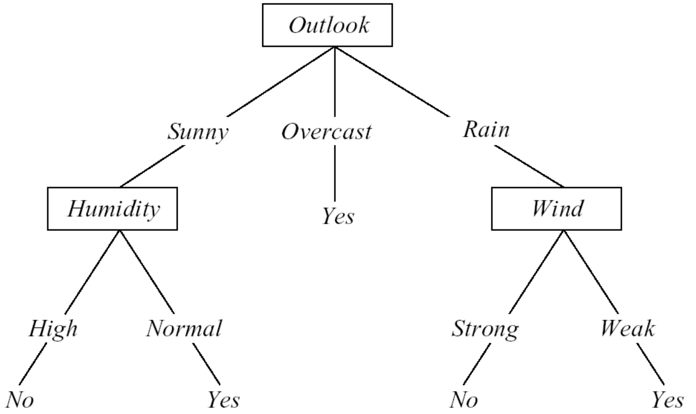
Decision Tree for *PlayTennis* (Example)



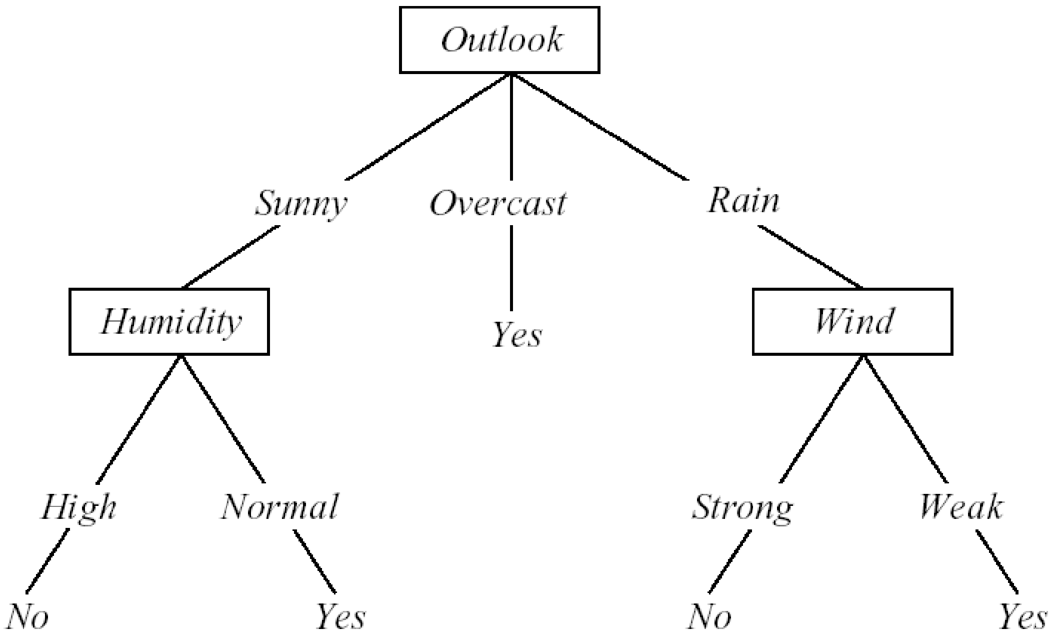
Decision Trees

- Decision tree representation:
  - Each internal node tests an attribute
  - Each branch corresponds to attribute value
  - Each leaf node assigns a classification

- How would we represent:
  - $\wedge$, $\vee$, XOR
  - $(A \wedge B) \vee (C \wedge \neg D \wedge E)$
  - M of N



Converting A Tree to Rules



IF $(Outlook = Sunny) \wedge (Humidity = High)$
THEN $PlayTennis = No$
IF $(Outlook = Sunny) \wedge (Humidity = Normal)$
THEN $PlayTennis = Yes$
….

A Tree to Predict Surgery (C-Section) Risk

- Learned from medical records of 1000 women
- Negative examples are C-sections

```
[833+,167-] .83+ .17-
Fetal_Presentation = 1: [822+,116-] .88+ .12-
| Previous_Csection = 0: [767+,81-] .90+ .10-
| | Primiparous = 0: [399+,13-] .97+ .03-
| | Primiparous = 1: [368+,68-] .84+ .16-
| | | Fetal_Distress = 0: [334+,47-] .88+ .12-
| | | | Birth_Weight < 3349: [201+,10.6-] .95+ .05
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+ .2
| | | Fetal_Distress = 1: [34+,21-] .62+ .38-
| Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
```

When to Consider Decision Trees

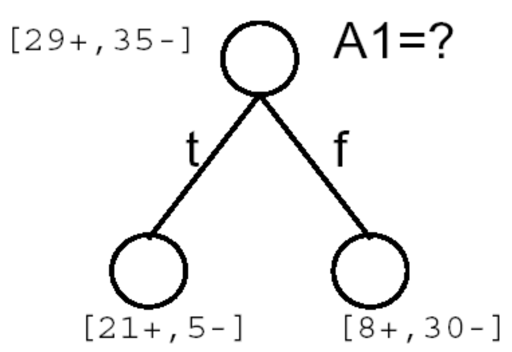
- Instances describable by attribute-value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data

Examples:

- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences

Top-Down Induction of Decision Trees (Approach)

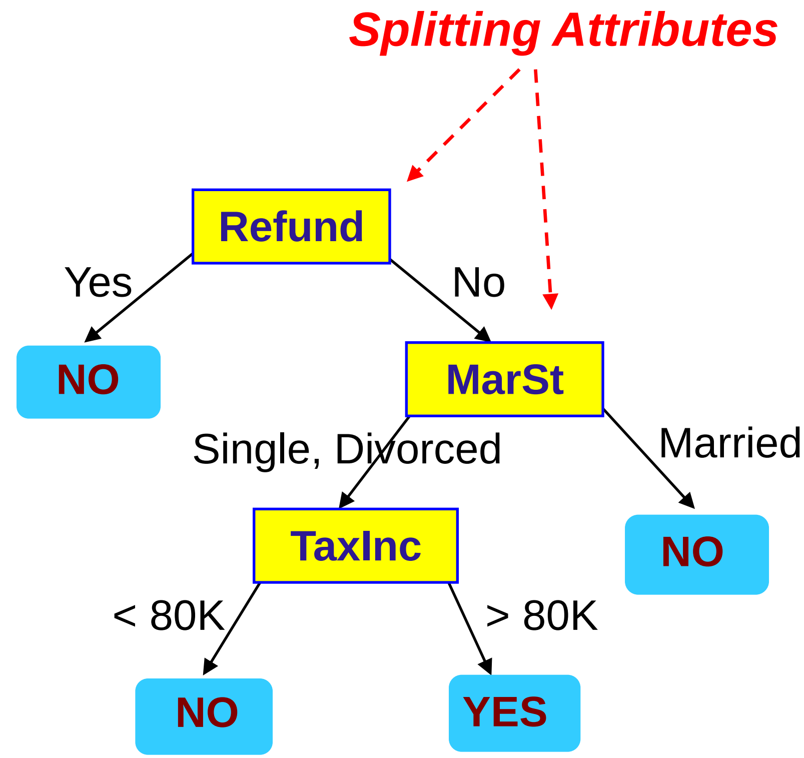
- Main loop:
  1. $A \leftarrow$ the "best" decision attribute for next *node*
  2. Assign A as decision attribute for *node*
  3. For each value of A, create new descendant of *node*
  4. Sort training examples to leaf nodes
  5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes
- Which attribute is best?



Example of a Decision Tree

	categorical	categorical	continuous	class
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

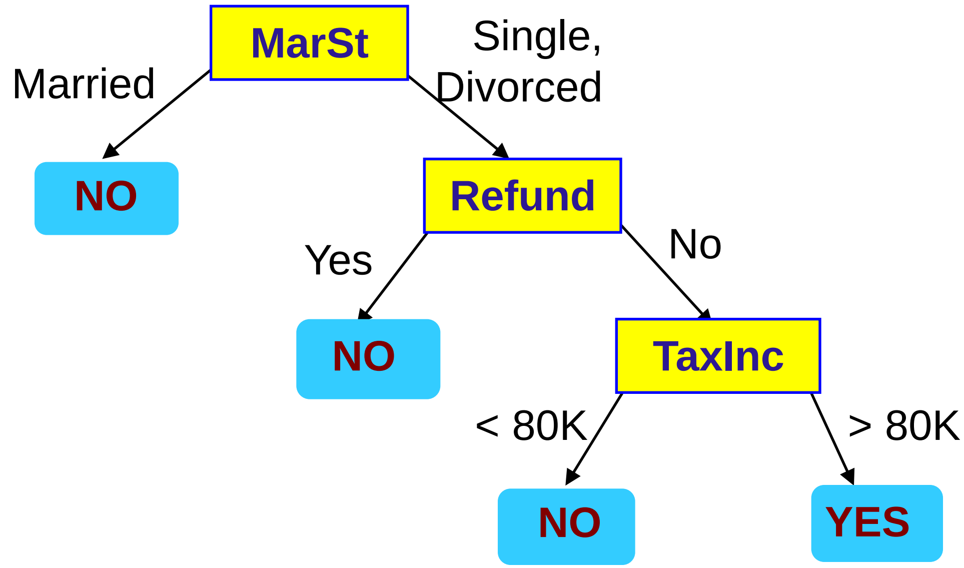


Model: Decision Tree

Another Example of Decision Tree

categorical categorical continuous class

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



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

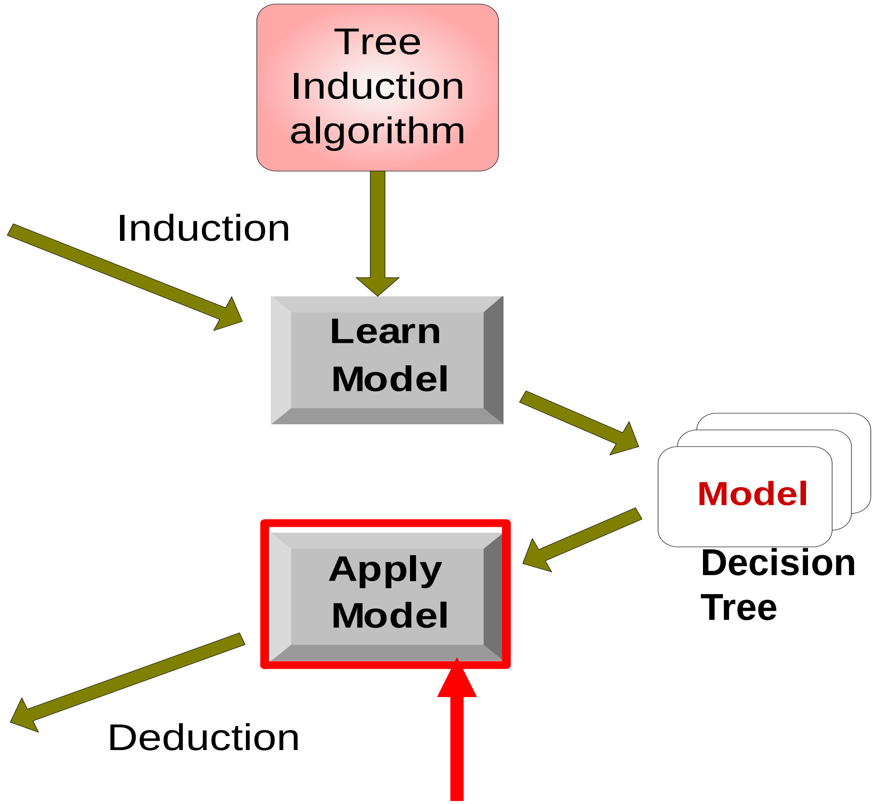
Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

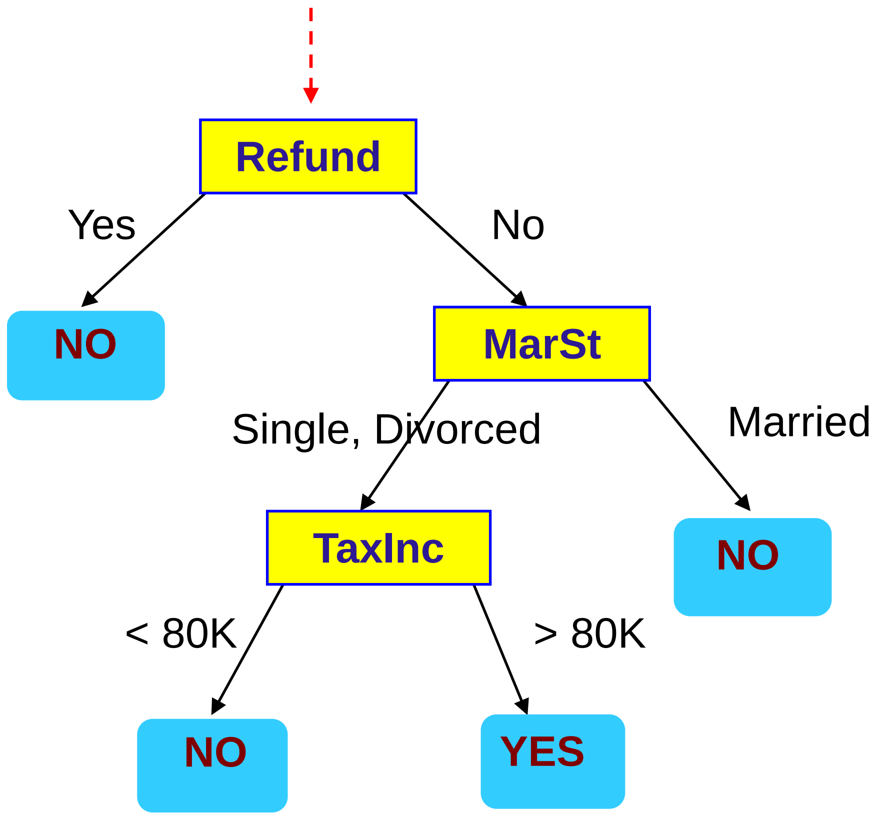
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Apply Model to Test Data

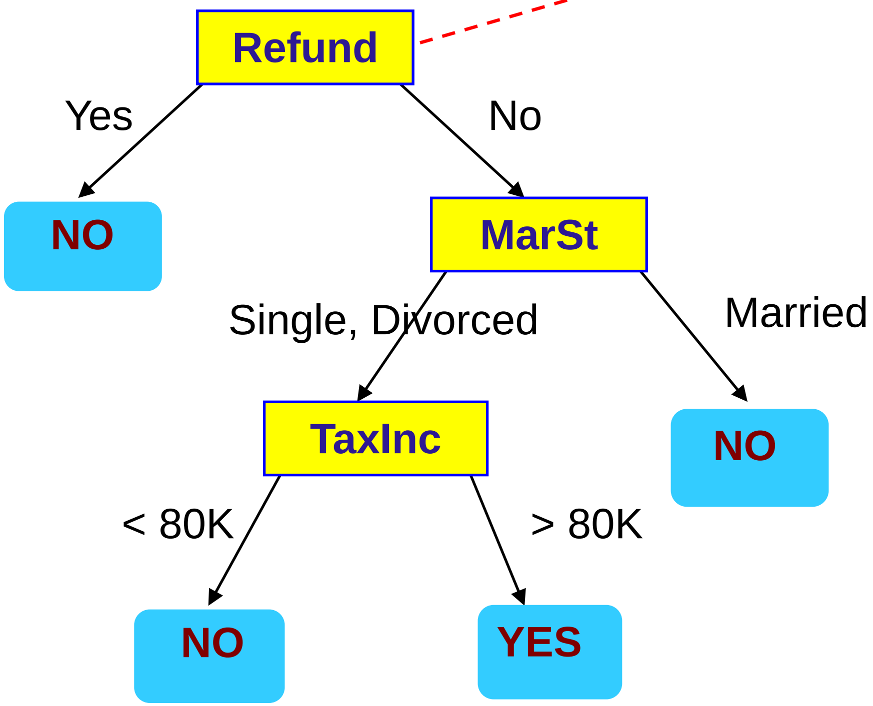
Start from the root of tree.



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

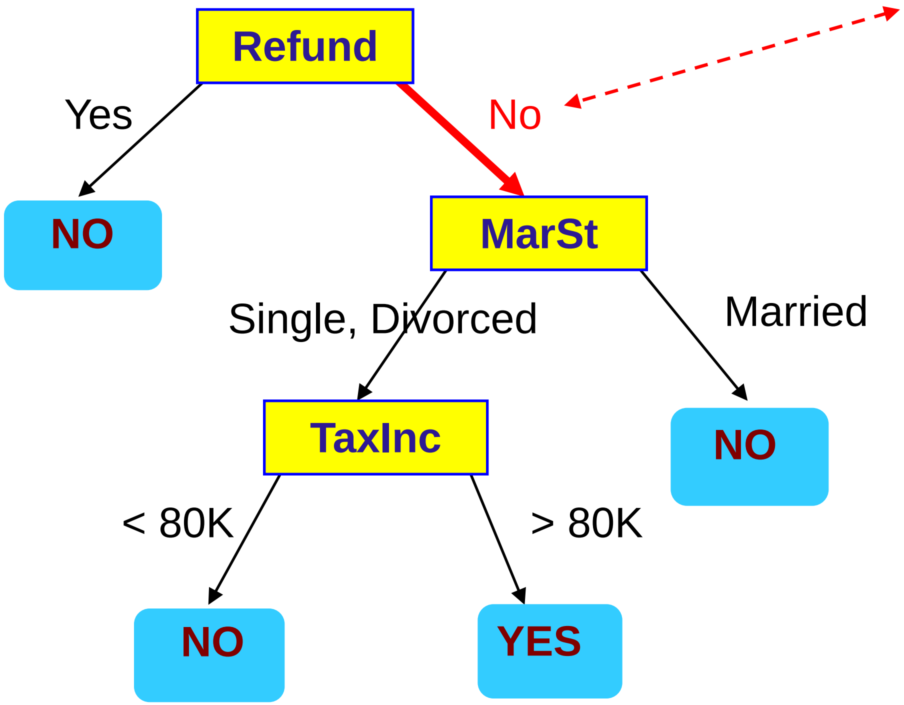
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	**?**

Apply Model to Test Data



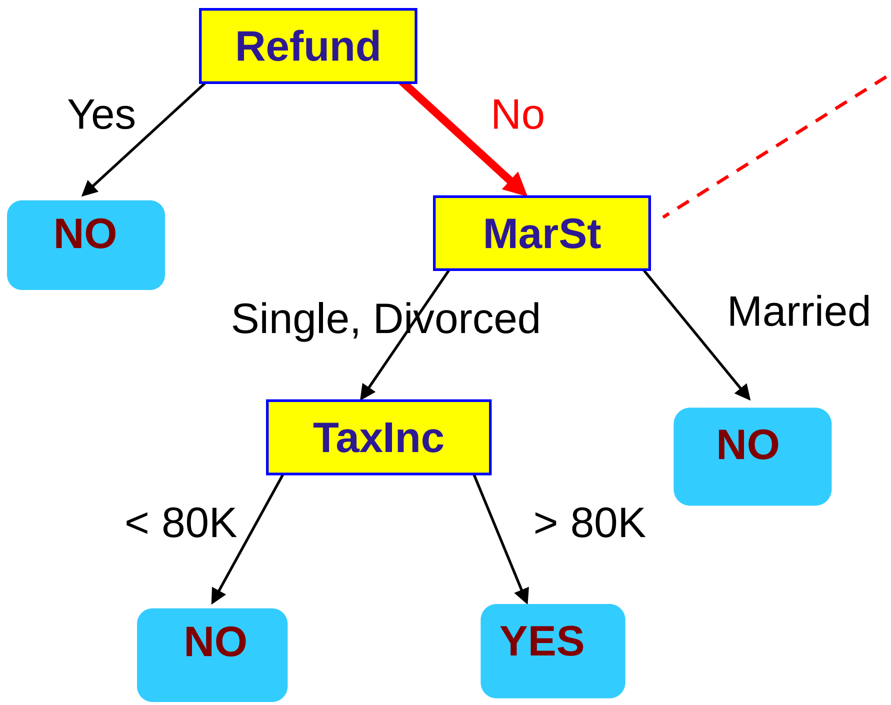
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

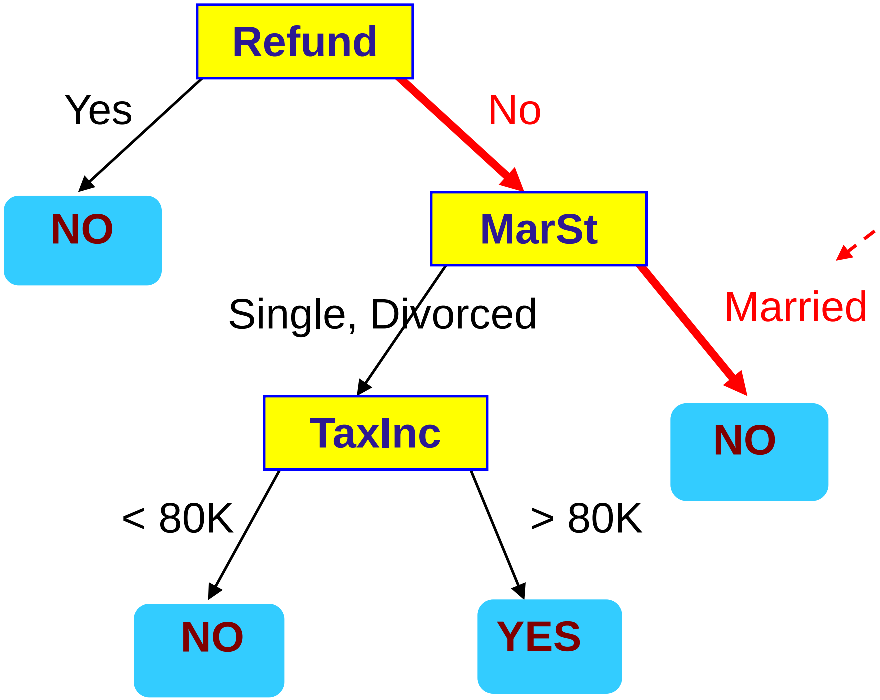
Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	**?**



Apply Model to Test Data



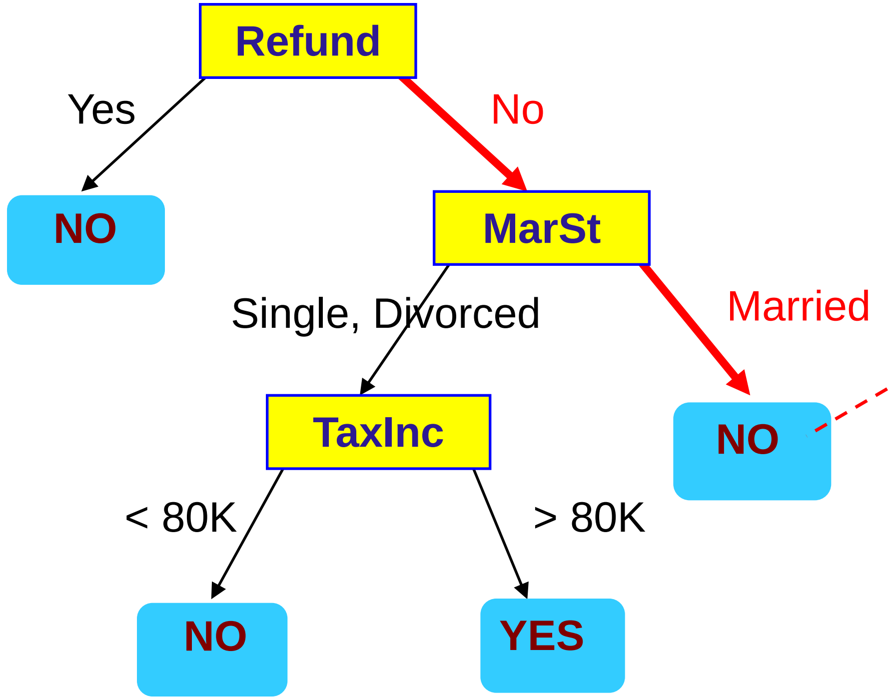
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

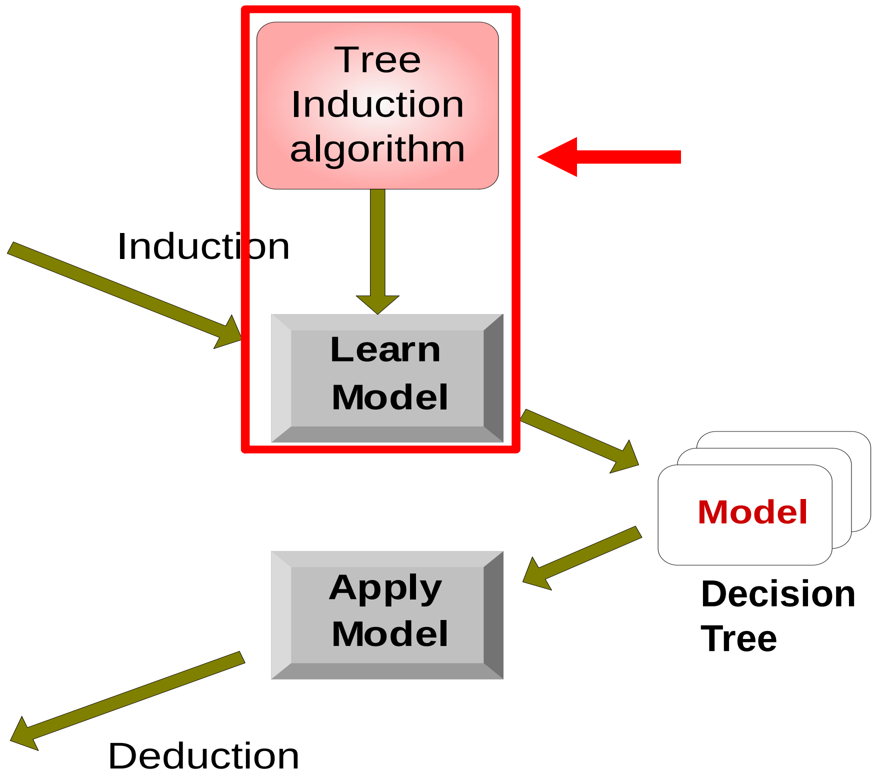


Assign Cheat to “No”

Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

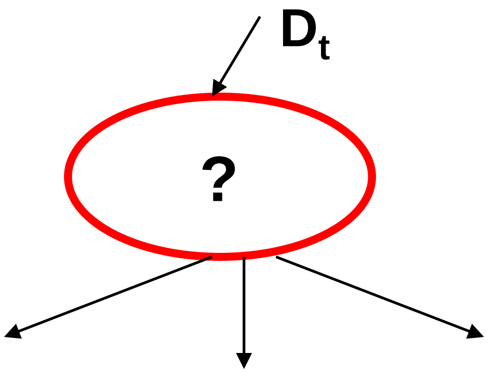
Decision Tree Induction

- Many Algorithms:
  - Hunt’s Algorithm (one of the earliest)
  - CART
  - ID3, C4.5
  - SLIQ, SPRINT

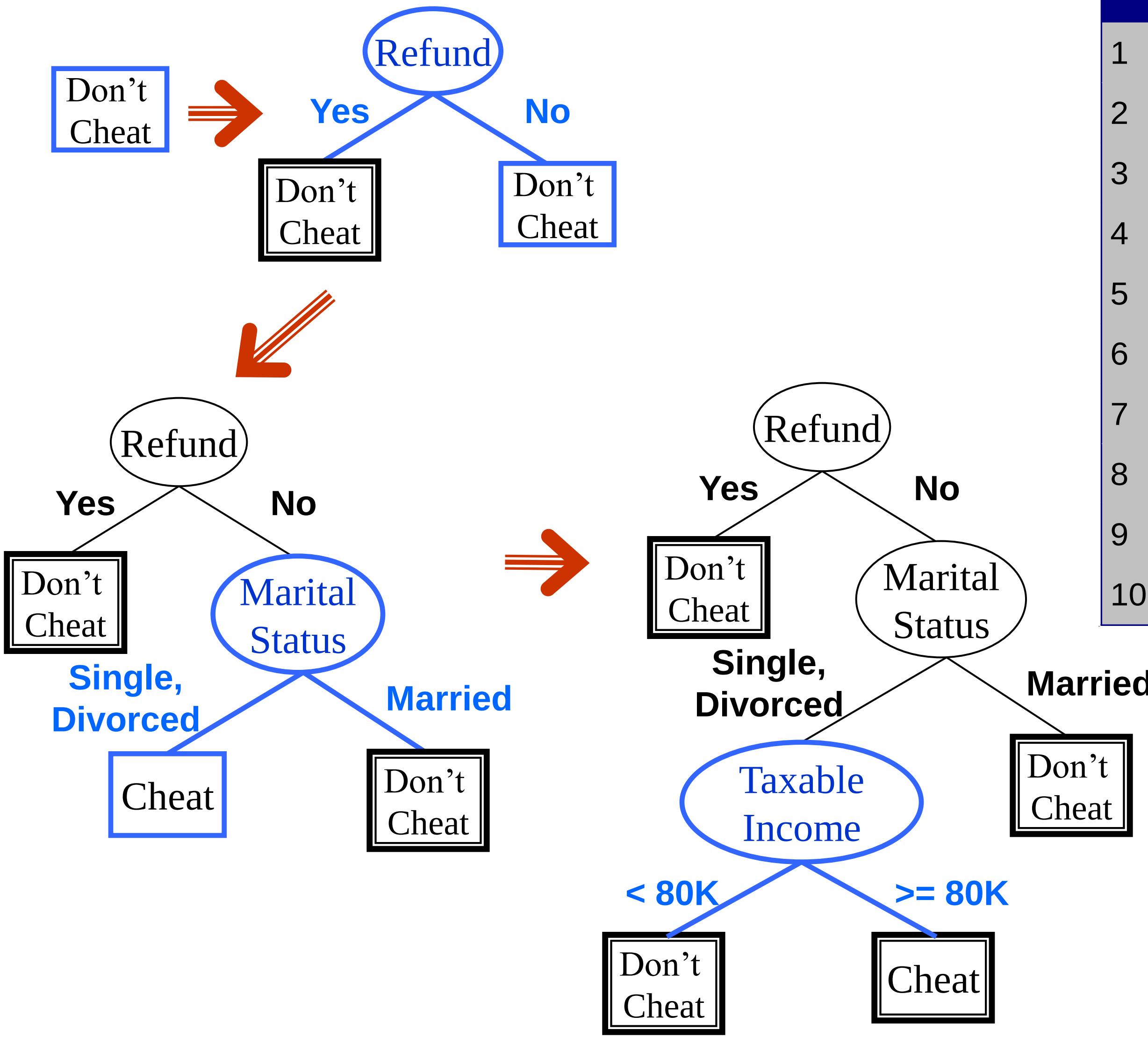
General Structure of Hunt's Algorithm

- Let D_t be the set of training records that reach a node t
- General Procedure:
  - If D_t contains records that belong the same class y_t, then t is a leaf node labeled as y_t
  - If D_t is an empty set, then t is a leaf node labeled by the default class, y_d
  - If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets. Recursively apply the procedure to each subset.

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



Hunt’s Algorithm



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

Tree Induction

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.
- Issues
  1. Determine how to split the records
     a) How to specify the attribute test condition?
     b) How to determine the best split?
  2. Determine when to stop splitting

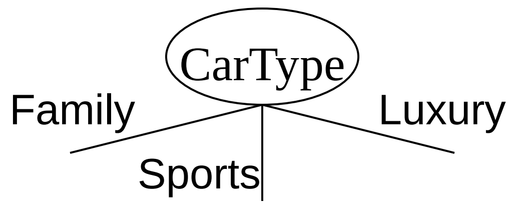
How to Split Records?

a) How to Specify Test Condition?

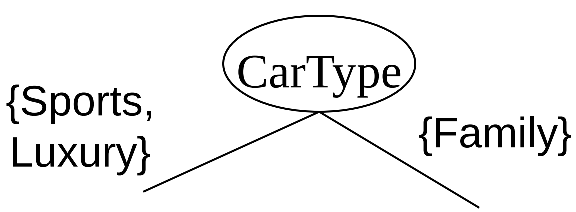
- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Splitting Based on Nominal Attributes

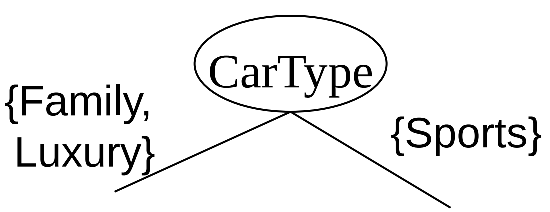
- Multi-way split: Use as many partitions as distinct values.



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

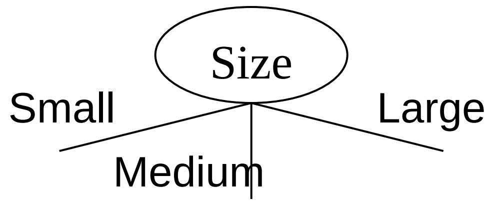


OR

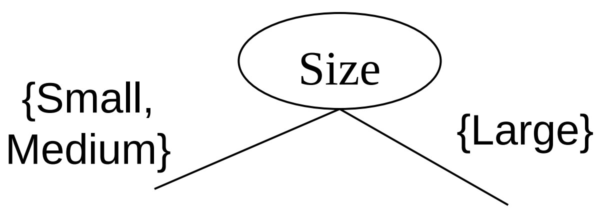


Splitting Based on Ordinal Attributes

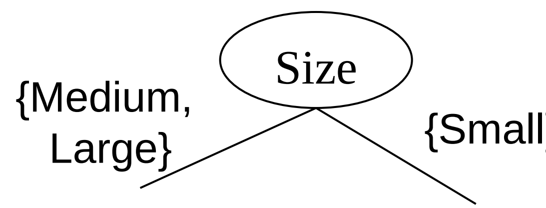
- Multi-way split: Use as many partitions as distinct values.



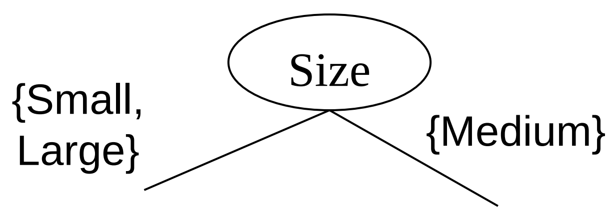
- Binary split: Divides values into two subsets.
  Need to find optimal partitioning.



OR



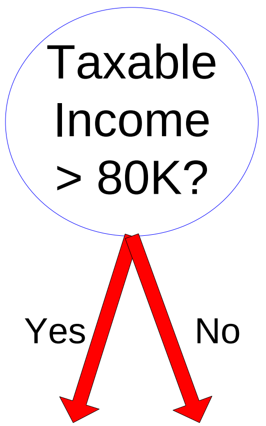
- What about this split?



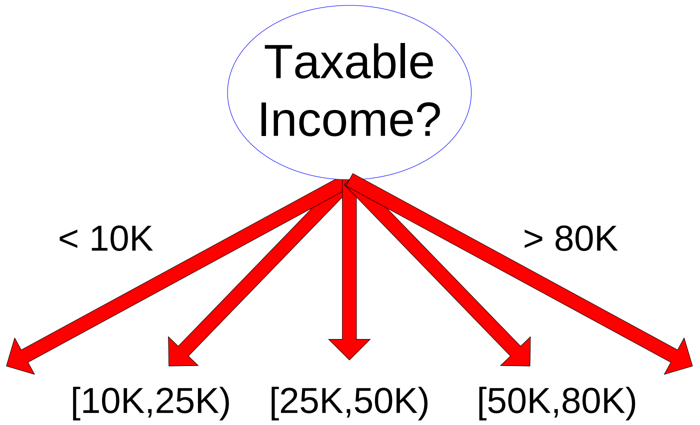
Splitting Based on Continuous Attributes

- Different ways of handling
  - Discretization to form an ordinal categorical attribute
    - Static – discretize once at the beginning
    - Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
  - Binary Decision: $(A < v)$ or $(A \geq v)$
    - consider all possible splits and finds the best cut
    - can be more compute intensive

Splitting Based on Continuous Attributes



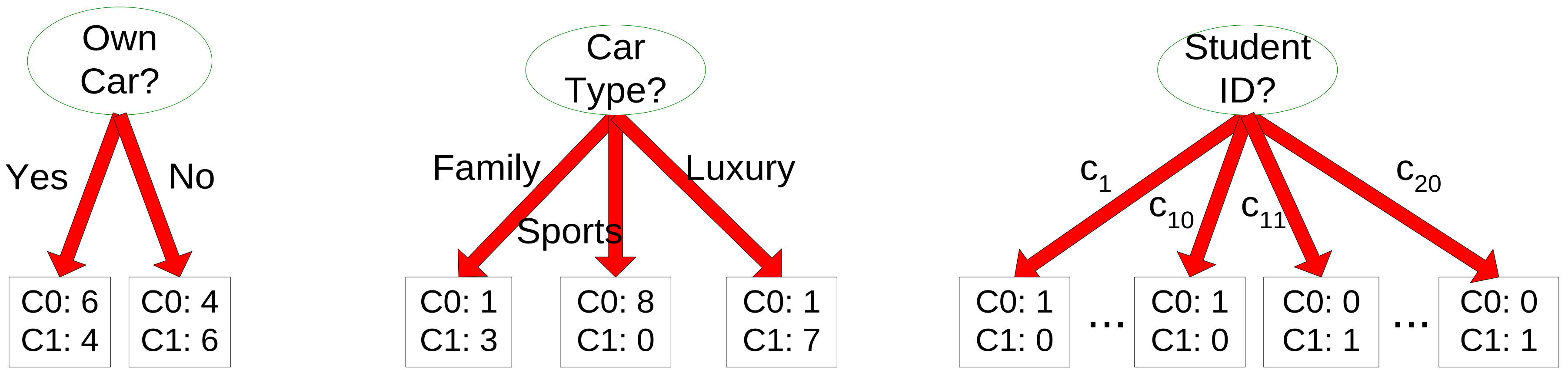
(i) Binary split



(ii) Multi-way split

b) How to determine the Best Split

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



Which test condition is the best?

How to determine the Best Split

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

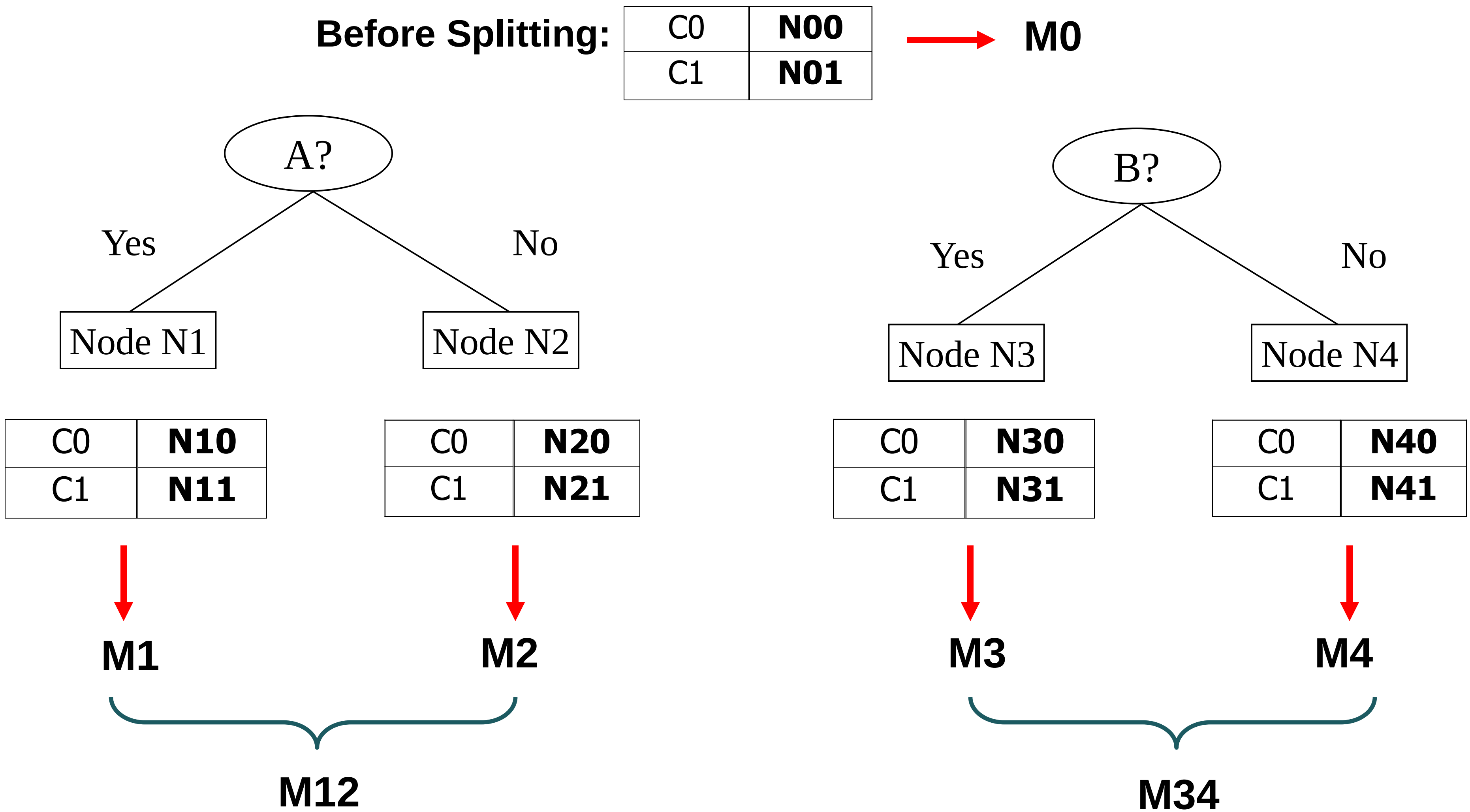
C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous, **High degree of impurity**	**Homogeneous,** **Low degree of impurity**

Measures of Node Impurity

Following two are the most commonly used method to measure node impurity:

1. Gini Index
2. Entropy

How to Find the Best Split



Gain = M0 – M12 vs M0 – M34

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 $(1 - 1/n_c)$ 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

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

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,

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

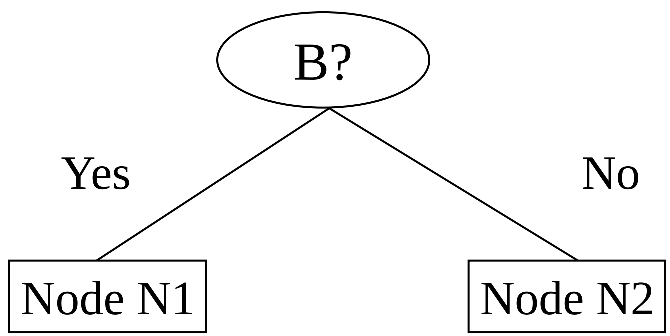
where, n_i = number of records at child i,

n = number of records at node p.

Binary Attributes: Computing GINI Index

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

	Parent
C1	6
C2	6
Gini = 0.500	



Gini(N1)
$= 1 - (5/6)^2 - (2/6)^2$
= 0.194

Gini(N2)
$= 1 - (1/6)^2 - (4/6)^2$
= 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

Categorical Attributes: Computing Gini Index

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

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	2	1
C2	4	1	1
Gini	0.393		

Two-way split
(find best partition of values)

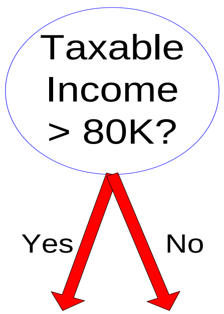
	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

	CarType	
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	0.419	

Continuous Attributes: Computing Gini Index

- Use Binary Decisions based on one value
- Several Choices for the splitting value
  - Number of possible splitting values = Number of distinct values
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions, A < v and A ≥ v
- Simple method to choose best v
  - For each v, scan the database to gather count matrix and compute its Gini index
  - Computationally Inefficient! Repetition of work.

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



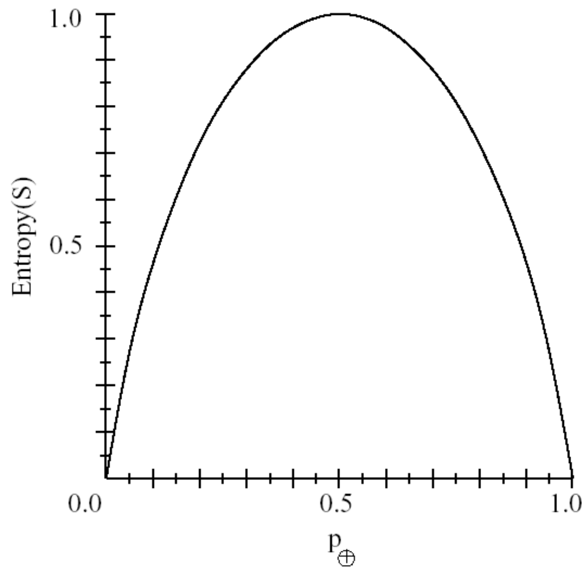
Continuous Attributes: Computing Gini Index...

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

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
	Taxable Income									
Sorted Values	60	70	75	85	90	95	100	120	125	220

Split Positions	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		*<u>0.300</u>*		0.343		0.375		0.400		0.420	

Entropy(1/2)



- S is a sample of training examples
- $p_\oplus$ is the proportion of positive examples in S
- $p_\ominus$ is the proportion of negative examples in S
- Entropy measures the impurity of S

$$Entropy(S) \equiv - p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$$

Entropy(2/2)

Entropy(*S*) = expected number of bits needed to encode class ($\oplus$ or $\ominus$) of randomly drawn member of S (under the optimal, shortest-length code)

Why?

Information theory: optimal length code assigns

- $\log_2 p$ bits to message having probability p.

So, expected number of bits to encode $\oplus$ or $\ominus$ of random member of *S*:

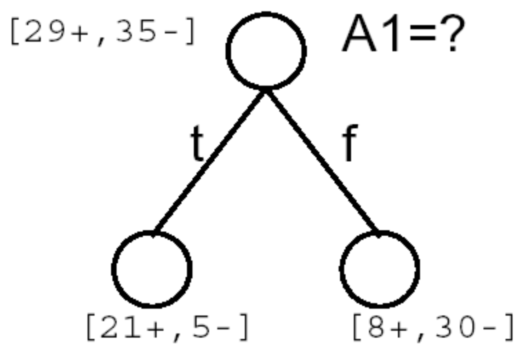
$$p_{\oplus}(-\log_2 p_{\oplus}) + p_{\ominus}(-\log_2 p_{\ominus})$$

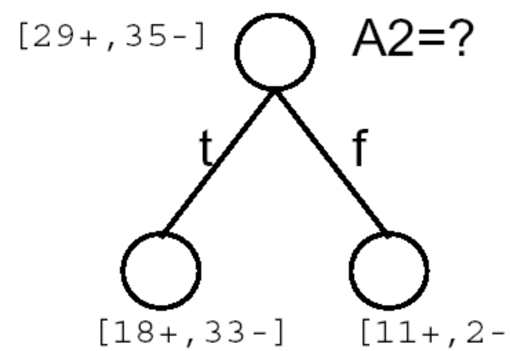
$$Entropy(S) \equiv - p_{\oplus}\log_2 p_{\oplus} - p_{\ominus}\log_2 p_{\ominus}$$

Information Gain

Gain(*S*, *A*) = expected reduction in entropy due to sorting on *A*

$$Gain(S,A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$



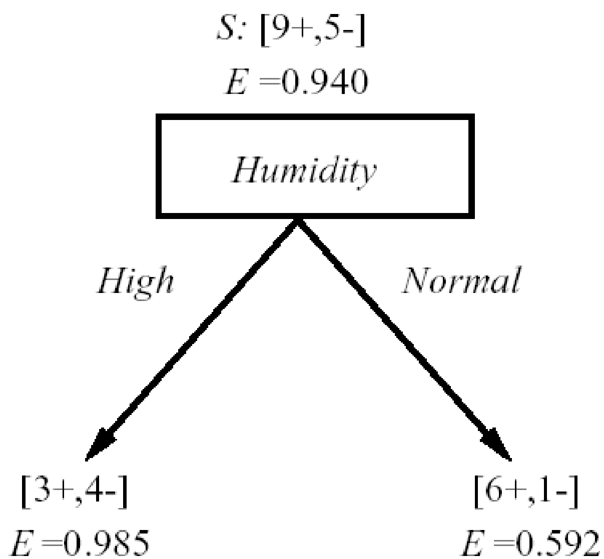


Training Examples

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

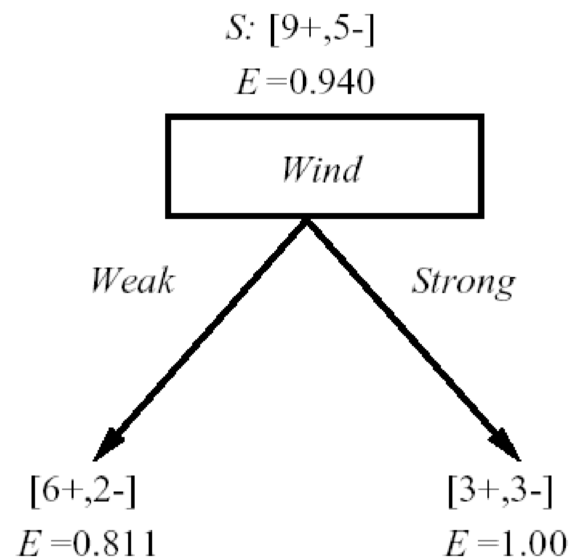
Selecting the Next Attribute(1/2)

Which attribute is the best classifier?



Gain (S, Humidity)

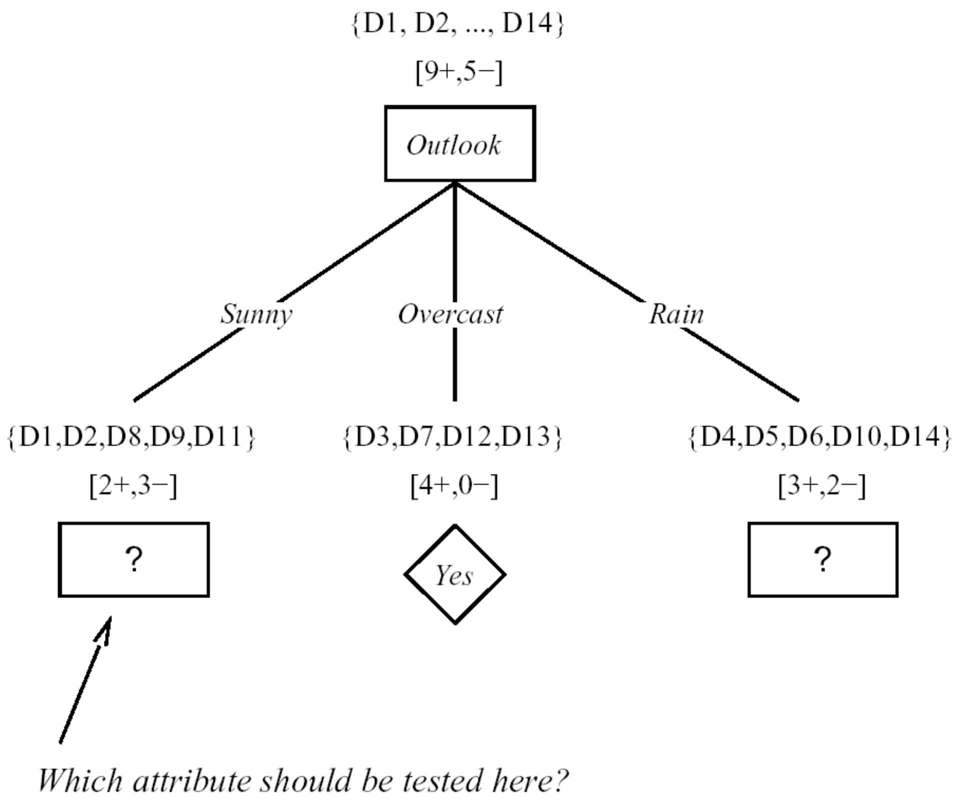
= .940 - (7/14).985 - (7/14).592
= .151



Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048

Selecting the Next Attribute(2/2)



S_{sunny} = {D1,D2,D8,D9,D11}

Gain (S_{sunny} *, Humidity)* = .970 - (3/5) 0.0 - (2/5) 0.0 = .970

Gain (S_{sunny} *, Temperature)* = .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .570

Gain (S_{sunny}*, Wind)* = .970 - (2/5) 1.0 - (3/5) .918 = .019

Alternative Splitting Criteria based on INFO

- Entropy at a given node t:

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

(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

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

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) $\log_2$ (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

Splitting Based on INFO...

- 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

– Measures Reduction in Entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)
– Used in ID3 and C4.5
– Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

Splitting Based on INFO...

- Gain Ratio:

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

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 Information Gain

Splitting Criteria based on Classification Error

- Classification error at a node t :

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

- Measures misclassification error made by a node.
  - Maximum ($1 - 1/n_c$) when records are equally distributed among all classes, implying least interesting information
  - Minimum (0.0) when all records belong to one class, implying most interesting information

Examples for Computing Error

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

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

Determine When to STOP Splitting

Stopping Criteria for Tree Induction

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

Decision Tree Based Classification

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

Example: C4.5

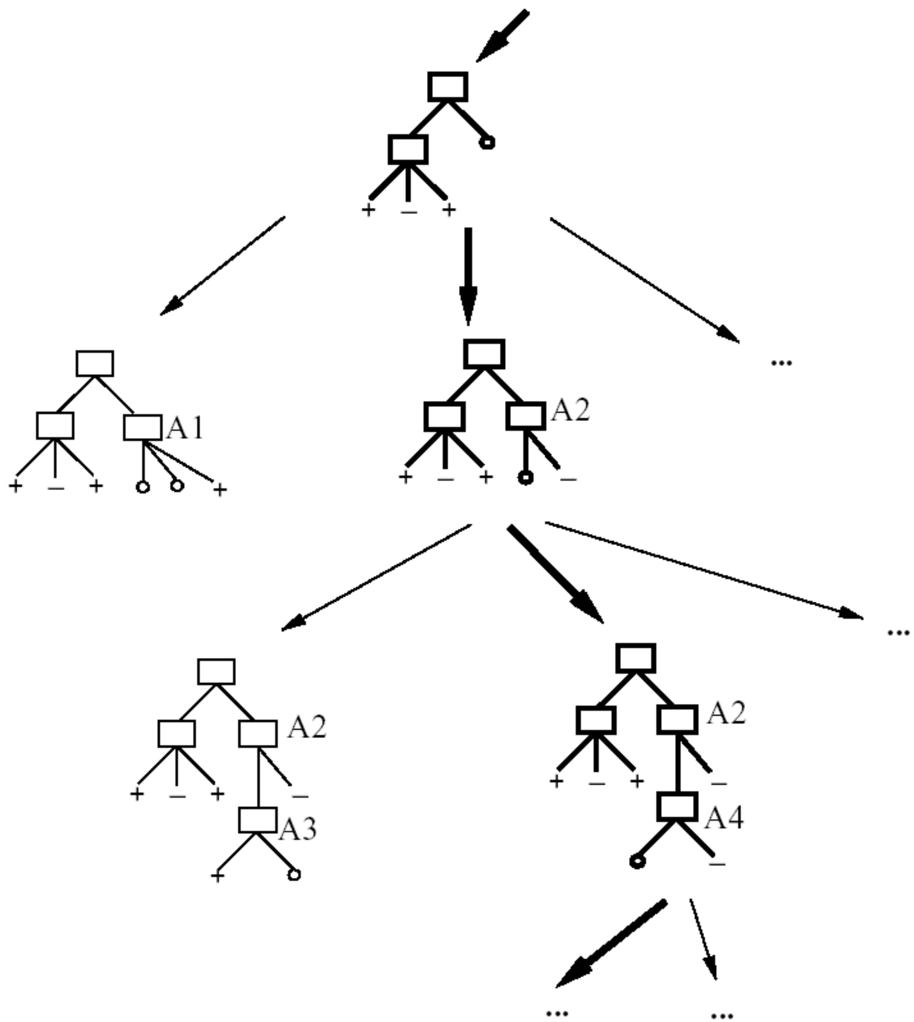
- Simple depth-first construction.
- Uses Information Gain
- Sorts Continuous Attributes at each node.
- Needs entire data to fit in memory.
- Unsuitable for Large Datasets.
  - Needs out-of-core sorting.

- You can download the software from: http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz

Practical Issues of Classification

1. Underfitting and Overfitting
2. Missing Values
3. Costs of Classification

Hypothesis Space Search by ID3(1/2)



Hypothesis Space Search by ID3(2/2)

- Hypothesis space is complete!
  - Target function surely in there...
- Outputs a single hypothesis (which one?)
  - Can't play 20 questions...
- No back tracking
  - Local minima...
- Statistically-based search choices
  - Robust to noisy data...
- Inductive bias: approx "prefer shortest tree"

Inductive Bias in ID3

Note H is the power set of instances X

→ Unbiased?

Not really...

- Preference for short trees, and for those with high information gain attributes near the root
- Bias is a *preference* for some hypotheses, rather than a *restriction* of hypothesis space H
- Occam's razor: prefer the shortest hypothesis that fits the data

Occam’s Razor

Why prefer short hypotheses?

Argument in favor :

- Fewer short hyps. than long hyps.
  - → a short hyp that fits data unlikely to be coincidence
  - → a long hyp that fits data might be coincidence

Argument opposed :

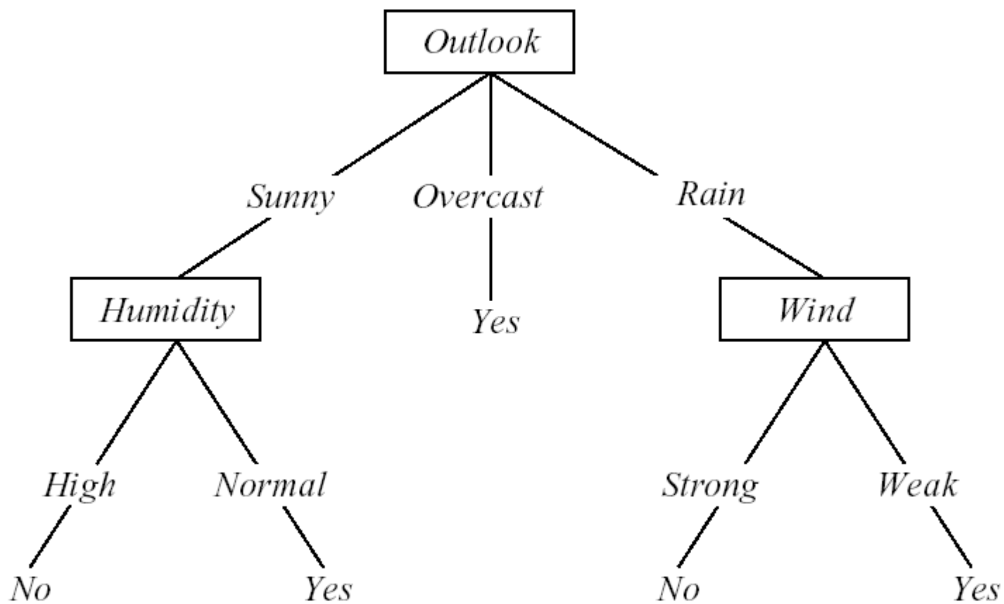
- There are many ways to define small sets of hyps
- e.g., all trees with a prime number of nodes that use attributes beginning with “Z”
- What's so special about small sets based on *size* of hypothesis??

Overfitting in Decision Trees

Consider adding noisy training example #15:

Sunny, *Hot*, *Normal*, *Strong*, *PlayTennis* = *No*

What effect on earlier tree?



Overfitting

Consider error of hypothesis h over

- training data: $error_{train}(h)$
- entire distribution D of data: $error_D(h)$

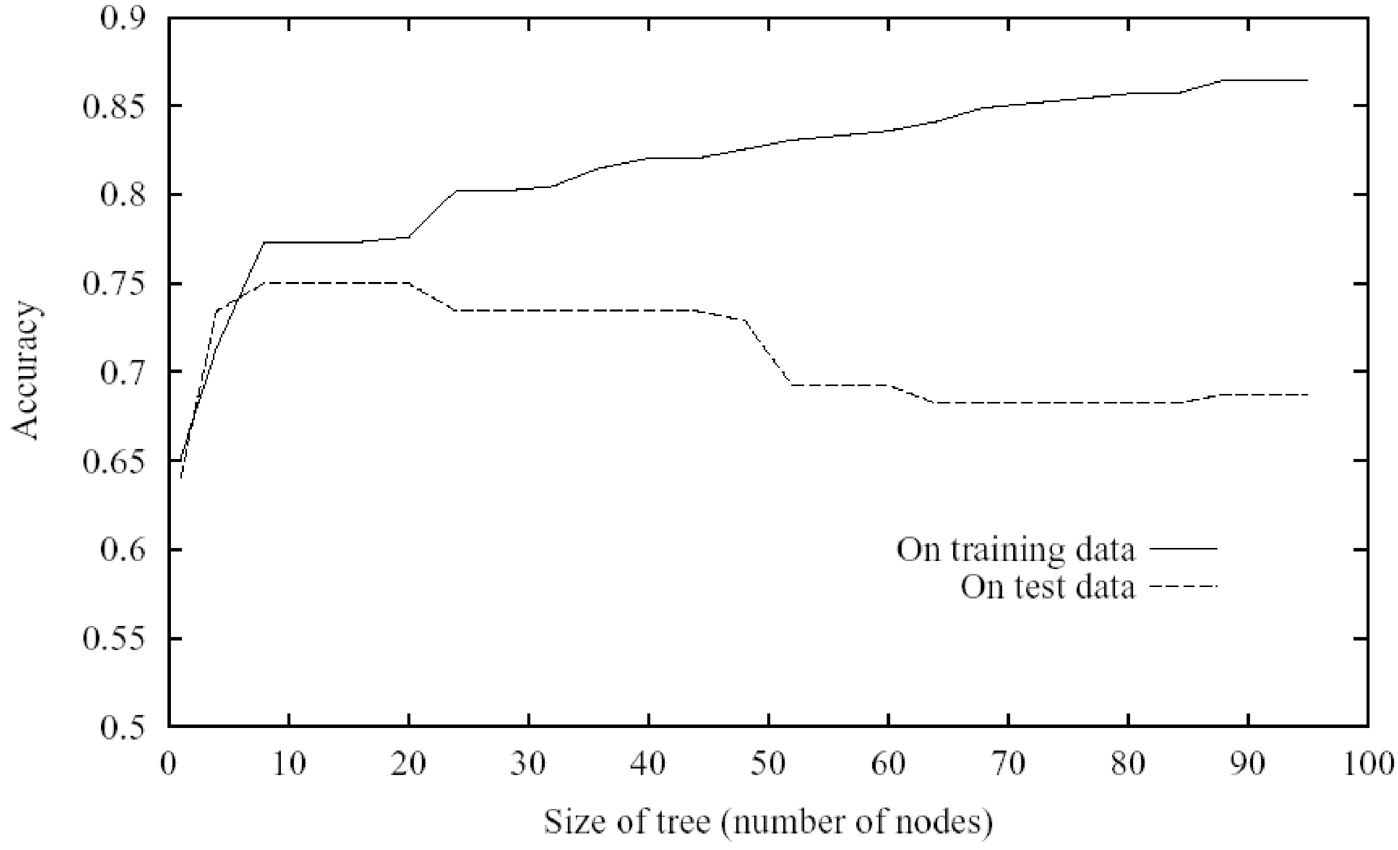
Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

$$error_{train}(h) < error_{train}(h')$$

and

$$error_D(h) > error_D(h')$$

Overfitting in Decision Tree Learning



Avoiding Overfitting

How can we avoid overfitting?

- stop growing when data split not statistically significant
- grow full tree, then post-prune

How to select "best" tree :

- Measure performance over training data
- Measure performance over separate validation data set
- MDL: minimize

  size(*tree*) + *size*(*misclassifications*(*tree*))

Reduced-Error Pruning

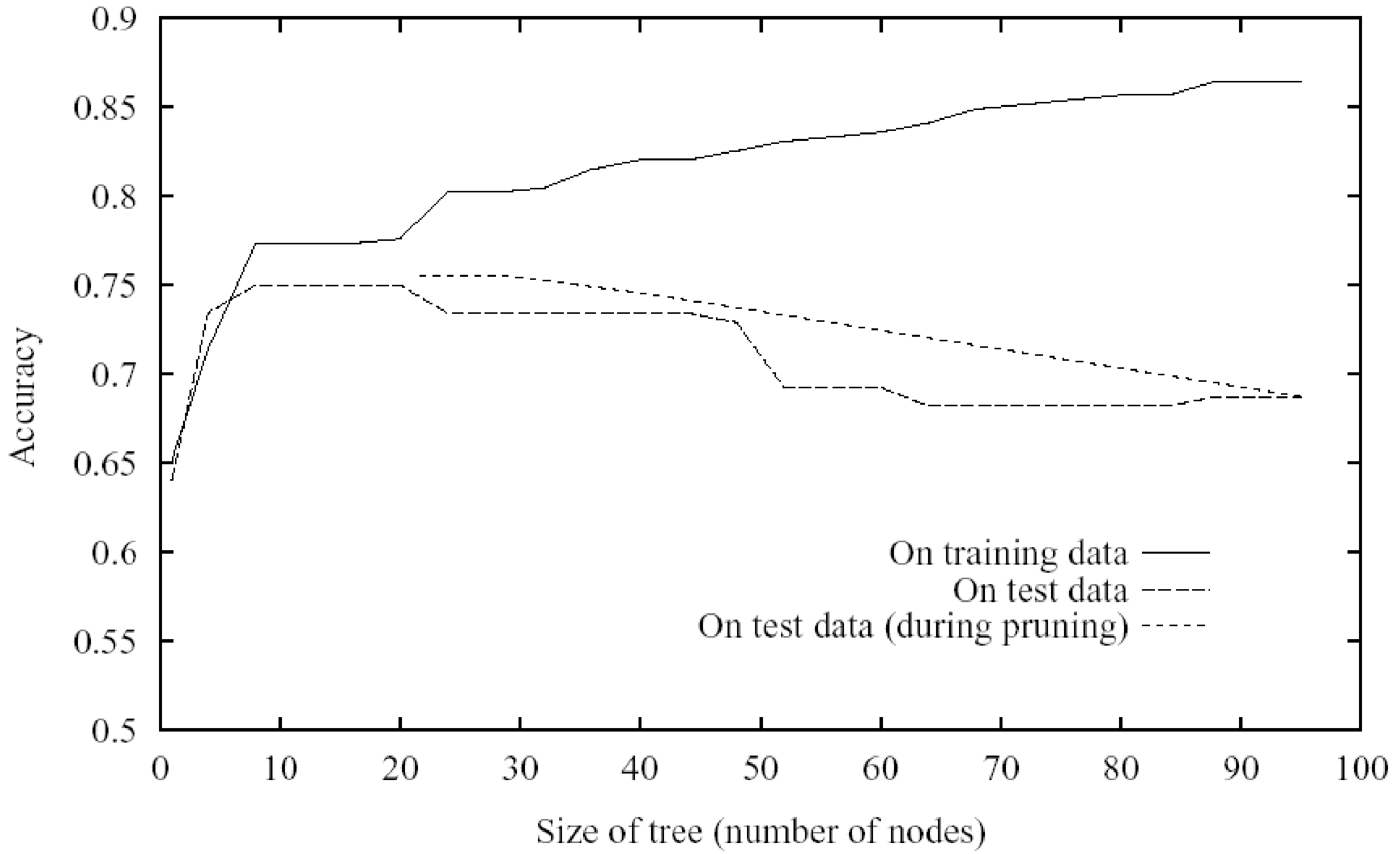
Split data into *training* and *validation* set

Do until further pruning is harmful:

1. Evaluate impact on *validation* set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves *validation* set accuracy

- produces smallest version of most accurate subtree
- What if data is limited?

Effect of Reduced-Error Pruning



Rule Post-Pruning

1. Convert tree to equivalent set of rules
2. Prune each rule independently of others
3. Sort final rules into desired sequence for use

Perhaps most frequently used method (e.g., C4.5)

Inferring Rudimentary Rules

Inferring Rudimentary Rules

- 1R: learns a 1-level decision tree
  - In other words, generates a set of rules that all test on one particular attribute
  - For example:
    - Outlook = sunny → play = yes
- Basic version (assuming nominal attributes)
  - One branch for each of the attribute's values
  - Each branch assigns most frequent class
  - Error rate: proportion of instances that don't belong to the majority class of their corresponding branch
  - Choose attribute with lowest error rate

Inferring Rudimentary Rules

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

- Goal: to classify on the column 'Play'
- Outlook=sunny → play = no
- Outlook=overcast → play = yes
- Outlook=rainy → play = yes

outlook	temperature	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	FASLE	yes
rainy	mild	high	TRUE	no

	attribute	rule			errors	total errors
1	outlook	sunny	-->	no	2/5	4/14
		overcast	-->	yes	0/4	
		rainy	-->	yes	2/5	

Inferring Rudimentary Rules

Attribute	Rules	Errors	Total errors
Outlook	Sunny → No	2/5	4/14
	Overcast → Yes	0/4	
	Rainy → Yes	2/5	
Temperature	Hot → No*	2/4	5/14
	Mild → Yes	2/6	
	Cool → Yes	1/4	
Humidity	High → No	3/7	4/14
	Normal → Yes	1/7	
Windy	False → Yes	2/8	5/14
	True → No*	3/6	

Statistical Modeling

Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

outlook	temperature	humidity	windy	play
sunny	cool	high	TRUE	?

Yes = 2/9*3/9*3/9*3/9*9/14=0.0053

No = 3/5*1/5*4/5*3/5*5/14=0.0206

Statistical Modeling

- Bayes's rules
- Yes = 2/9*3/9*3/9*3/9*9/14=0.0053
- No = 3/5*1/5*4/5*3/5*5/14=0.0206

Likelihood of the two classes

For "yes" = $2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0053$

For "no" = $3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0206$

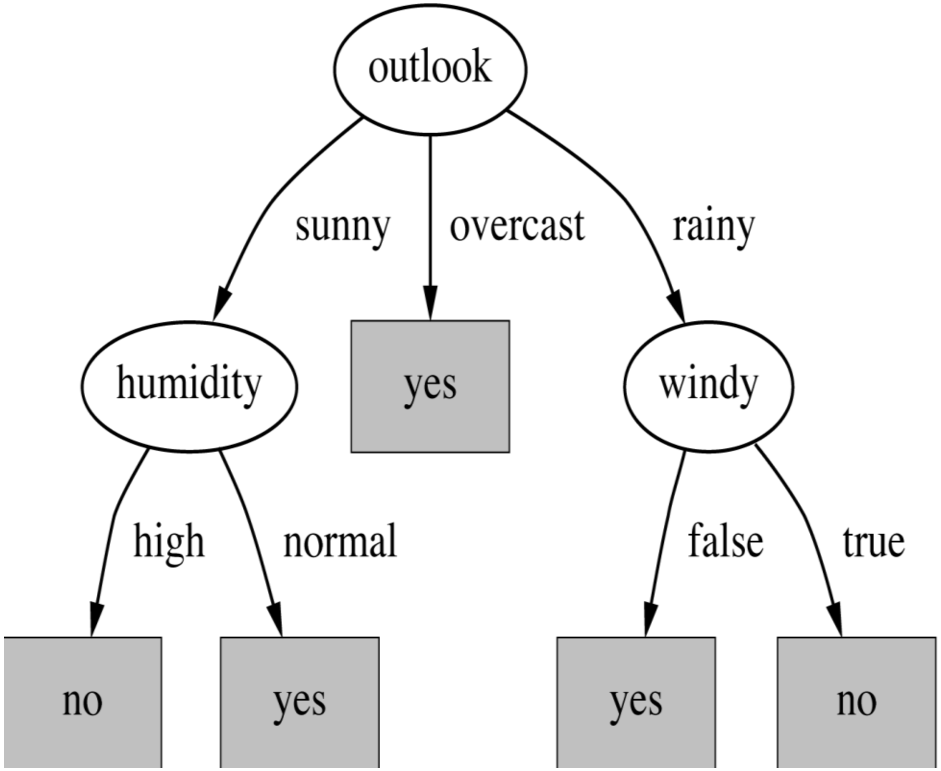
Conversion into a probability by normalization:

P("yes") = 0.0053 / (0.0053 + 0.0206) = 0.205

P("no") = 0.0206 / (0.0053 + 0.0206) = 0.795

Divide and Conquer: Constructing Decision Trees

outlook	temperature	humidity	windy	play
sunny	85	85	FALSE	no
sunny	80	90	TRUE	no
overcast	83	86	FALSE	yes
rainy	70	96	FALSE	yes
rainy	68	80	FALSE	yes
rainy	65	70	TRUE	no
overcast	64	65	TRUE	yes
sunny	72	95	FALSE	no
sunny	69	70	FALSE	yes
rainy	75	80	FALSE	yes
sunny	75	70	TRUE	yes
overcast	72	90	TRUE	yes
overcast	81	75	FASLE	yes
rainy	71	91	TRUE	no



Which attribute to select ?

Divide and Conquer: Constructing Decision Trees

- Which is the best attribute?
  - The one which will result in the smallest tree
- Popular impurity criterion: information gain
  - Information gain increases with the average purity of the subsets that an attribute produces
- Strategy: choose attribute that results in greatest information gain

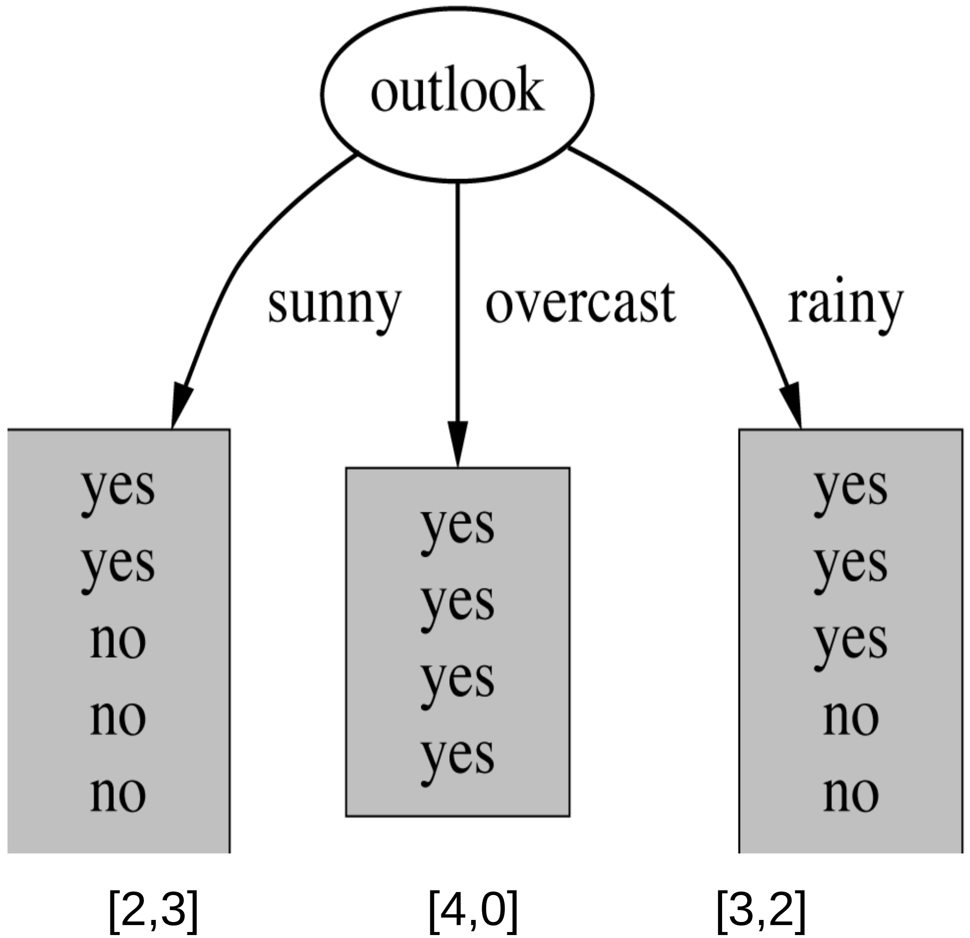
Divide and Conquer: Constructing Decision Trees

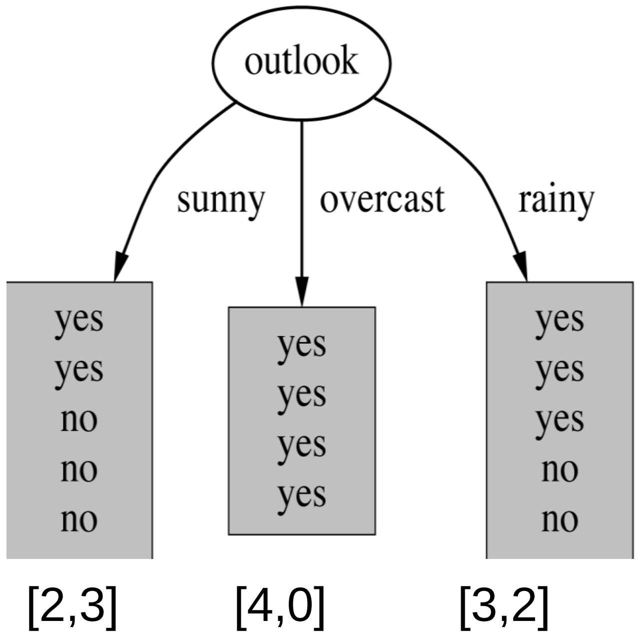
- Information is measured in bits
  - Given a probability distribution, the info required to predict an event is the distribution's entropy
  - Entropy gives the information required in bits (this can involve fractions of bits!)
- Formula for computing the entropy:

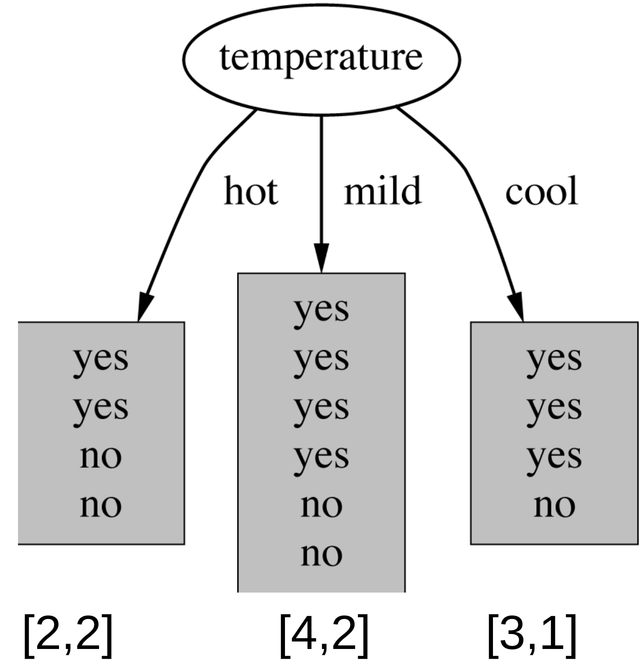
$$\text{entropy}(p_1, p_2, \ldots, p_n) = -p_1 \log p_1 - p_2 \log p_2 \ldots - p_n \log p_n$$

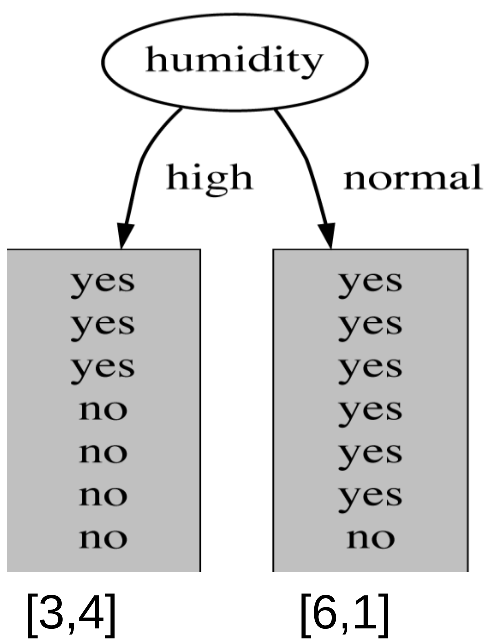
Divide and Conquer: Constructing Decision Trees

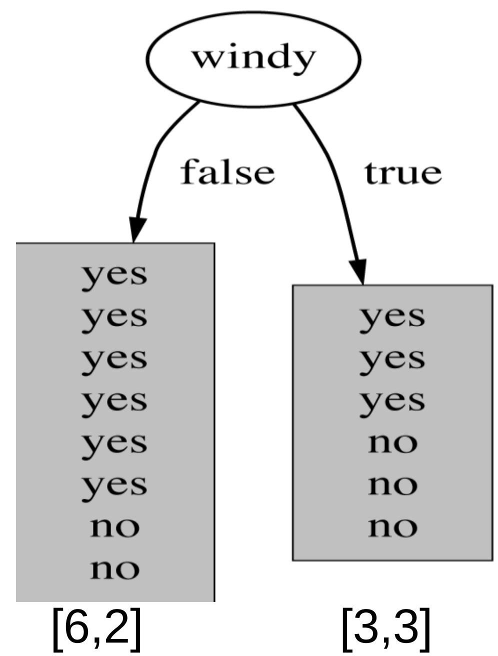
outlook	play
sunny	no
sunny	no
overcast	yes
rainy	yes
rainy	yes
rainy	no
overcast	yes
sunny	no
sunny	yes
rainy	yes
sunny	yes
overcast	yes
overcast	yes
rainy	no




outlook
sunny
overcast
rainy
yes yes no no no
yes yes yes yes
yes yes yes no no
[2,3]
[4,0]
[3,2]


temperature
hot
mild
cool
yes yes no no
yes yes yes yes no no
yes yes yes no
[2,2]
[4,2]
[3,1]

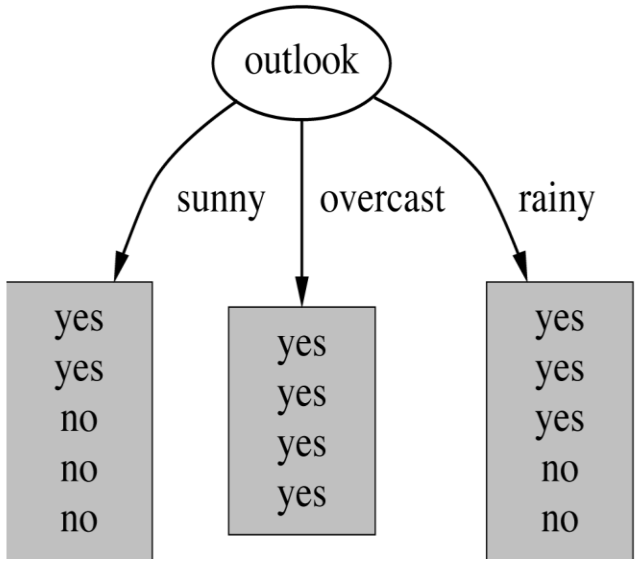

humidity
high
normal
yes yes yes no no no no
yes yes yes yes yes yes no
[3,4]
[6,1]


windy
false
true
yes yes yes yes yes yes no no
yes yes yes no no no
[6,2]
[3,3]

Divide and Conquer: Constructing Decision Trees

- – The number of either yeses or nos is zero, the information is zero
- – The number of yeses and nos is equal, the information reaches a maximum

Divide and Conquer: Constructing Decision Trees

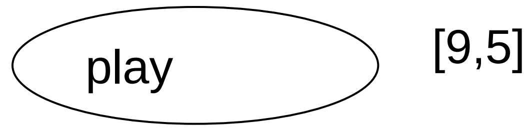


[2,3] [4,0] [3,2]

- Info ([2,3]) = -2/5 * log 2/5 – 3/5 * log 3/5 =0.971
- Info ([4,0]) = -4/4 * log 4/4 – 0/4 * log 0/4 =0
- Info ([3,2]) = -3/5 * log 3/5 – 2/5 * log 2/5 =0.971
- Info ([2,3],[4,0],[3,2])

  = 5/14 *0.971 + 4/14 *0 + 5/14 *0.971 = 0.673 bits

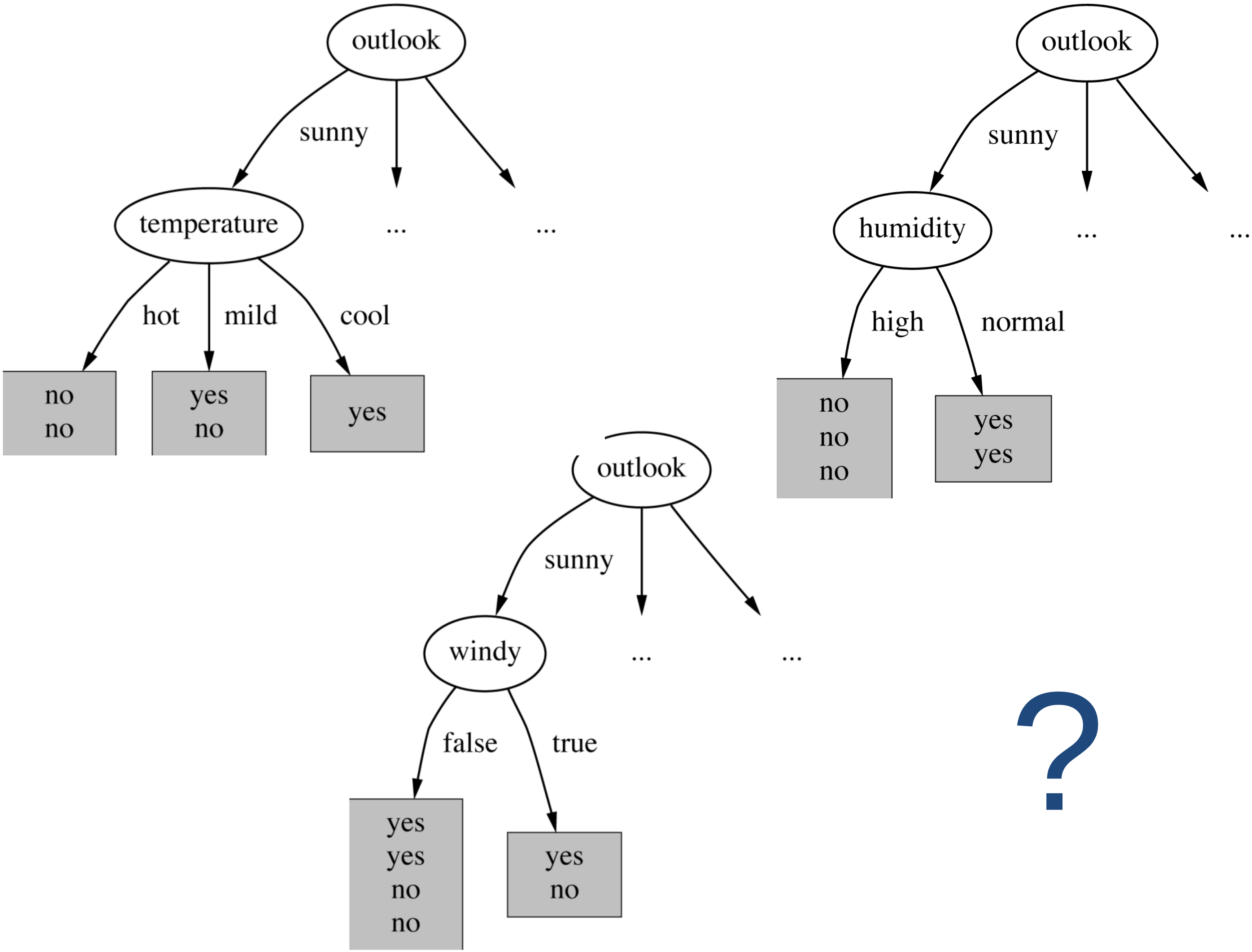
Divide and Conquer: Constructing Decision Trees



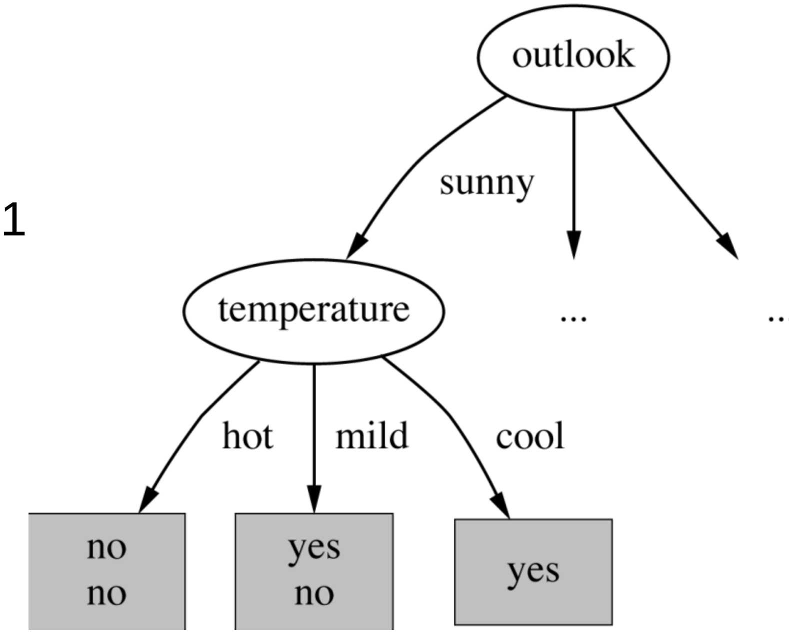
- Play: Info ([9,5]) = -9/14 * log 9/14 – 5/14 * log 5/14

  = 0.94 bits
- Gain (outlook) = Info ([9,5]) - Info ([2,3],[4,0],[3,2])

  = 0.94 - 0.673 = 0.247 bits
- Gain (temperature) = 0.029 bits
- Gain (humidity) = 0.029 bits
- Gain (windy) = 0.048 bits


outlook
sunny
temperature
...
...
hot
mild
cool
no
no
yes
no
yes
outlook
sunny
humidity
...
...
high
normal
no
no
no
yes
yes
outlook
sunny
windy
...
...
false
true
yes
yes
no
no
yes
no
?

Info ([2,3]) = 0.971

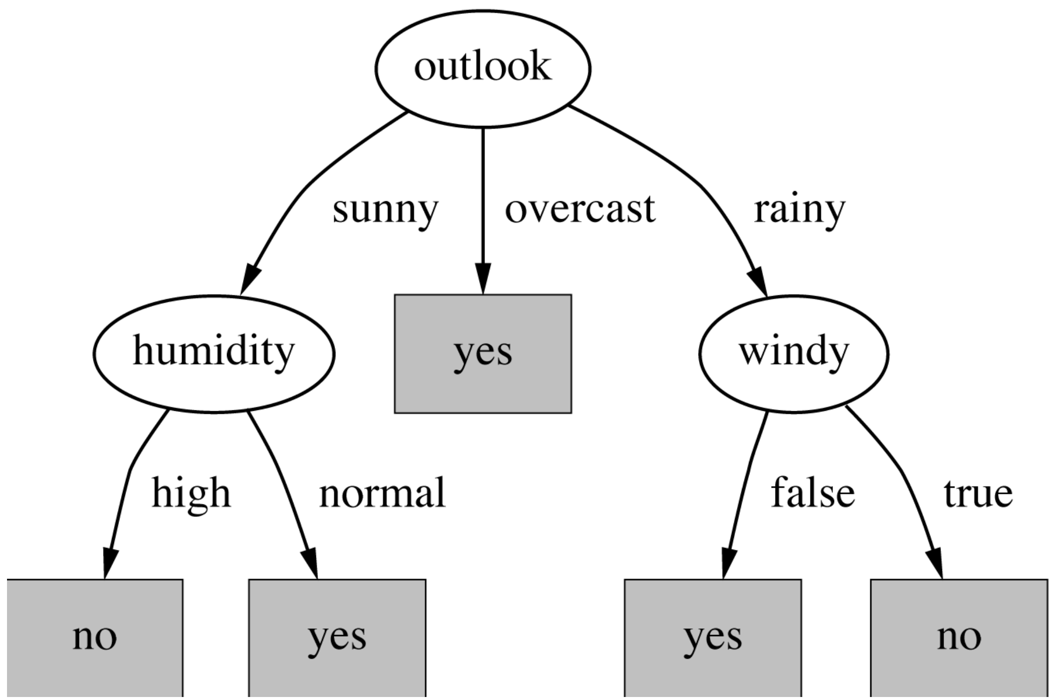


[0,2] [1,1] [1,0]

- Info ([0,2]) = 0
- Info ([1,1]) = -1/1 * log 1/1 – 1/1 * log 1/1
- Info ([1,0]) = 0
- Info ([0,2], [1,1], [1,0]) = 0.4 bits
- Gain (temperature) = 0.971 – 0.4 = 0.571 bits

Divide and Conquer: Constructing Decision Trees

- Gain (temperature) = 0.571 bits
- Gain (humidity) = 0.971 bits
- Gain (windy) = 0.020 bits



Lab Project Session (Weka)