[Placeholder for iClicker]

CPSC 100

Computational Thinking

Algorithm, Classifiers and Trees!

Instructor: Parsa Rajabi
Department of Computer Science
University of British Columbia

Agenda

- Course Admin
- Classifiers
- Decision Trees
- Entropy

Learning Goals

Learning Goals

After this lecture, you should be able to:

- Describe the **classification** steps.
- Explain the concept of a **rooted tree** and **decision tree**.
- Describe what the general decisions are in building a decision tree.
  - **Build a decision tree using entropy.**
- Describe **what considerations** are important in building a decision tree.

Course Admin

Algorithms

Algorithms

Recal
l

An ***algorithm*** describes a sequence of steps that is:

1. Unambiguous
   - No "assumptions" are required to execute the algorithm
   - The algorithm uses precise instructions
2. Executable
   - The algorithm can be carried out in practice
3. Terminating
   - The algorithm will eventually come to an end, or halt

Classifiers

Classifier

- A **classifier** is an **algorithm** that maps the input data to a specific category
  - Classifiers are derived from patterns or correlations from data.

Classifier: Training vs Test Data

- The data that classifiers learn the patterns has the "answer"
  - This data is called **training data**.

- Some of the training data is held back to check and see if the classifier works.
  - This is called **test data.**

Classifiers + Data

Classifiers then apply these patterns to new data with no “answer”

- **Input:** Digital image
- **Output:** Cat/not a cat
- **Training data:**
  Labeled images of cats and images that are not cats

Classification Task: Loans

Classification Task - Cancer Treatment

Input: Genome sequence from cancerous biopsy tissue

- Address, age, gender, credit rating, etc.

Output: Which cancer treatment is likely to work best

Training data:

Labeled genome sequences and which treatments were successful from both cancerous tissue

Classification Task - Loan Applications

Input: Individual's loan application

- Address, age, gender, credit rating, etc.

Output: Acceptance/Rejection

Training data:

List of loan apps, decisions made, and for those who were approved, whether they repaid the loan or not

Building a Classifier: Loans

Building a Classifier: Loans

You want to create a classifier to help you decide whether or not to give people loans

Here is your past (training) data on some loans

Applicant	Income	Gave loan?
#1	$50,005	Yes
#2	$25,004	No
#3	$75,005	Yes
#4	$95,005	Yes
#5	$45,007	No

Task: *Create an algorithm to decide what your classifier does: i.e., when will you give a loan, and when will you not give a loan?*

[Placeholder for iClicker]

Classification

Classification

The idea behind classification is that we want to use patterns and/or correlations to make decisions

Classification happens all the time in real life

- The doctor uses your symptoms and other measurements like weight/blood pressure/etc. to help make a diagnosis
- Google uses classification to determine what an image is

Classification is a general class of algorithms.

Classifiers

Classifiers are algorithms that perform classification

They are specific

- e.g. - we don't give loans to anyone an income of less than $50,000 per year

The algorithm you come up with is no different than the other algorithms you've come up with so far

You still need to state the steps you need to take to come up with the solution

Steps to do Classification

Step 1: Start with the data you have

Applicant	Annual Income	Loan Approved?
#1	26 000	No
#2	60 000	Yes
#3	50 000	Yes
#4	47 000	No
#5	12 000	No
#6	108 000	Yes

Step 2: Split data into training and test sets

We chose a 50/50 split for our demo but you could do other splits like 60/40, 70/30. For large datasets, 80/20 split is used

Training Data

Applicant	Annual Income	Loan Approved?
#1	26 000	No
#2	60 000	Yes
#3	50 000	Yes

Test Data

Applicant	Annual Income	Loan Approved?
#4	47 000	No
#5	12 000	No
#6	108 000	Yes

Step 3: Build classifier
(i.e., Find pattern in training set)

Given your training data, can you find a pattern that can tell you when to approve a loan?

Earlier, we decided an annual income of ~$50,000 seemed Like a good cut off point. **That was a classifier!**

Applicant	Annual Income	Loan Approved?
#1	26 000	No
#2	60 000	Yes
#3	50 000	Yes

Training Data

Step 4: Use classifier on test data

Applicant	Annual Income	Loan Approved?
#4	47 000	?
#5	12 000	?
#6	108 000	?

Test Data

After you come up with a classifier that seems to do okay with your training data, you use it on your test data to see what kinds of decisions it makes.

Step 5: Calculate Accuracy

Applicant	Annual Income	Loan Approved?	Classifier said to...
#4	47 000	No	No
#5	12 000	No	No
#6	108 000	Yes	Yes

Test Data

If the results of your classifier match up with the decisions you've made in your test data, it's looking good.

You can start trying to use it on data that you haven't made any decisions on yet.

Seems Straightforward

- What happens when we have more than one attribute?
- In the example before, we only had to consider annual income
- But what would happen if we had multiple attributes, like 5 or 10 or 100?
- **How do we decide which attribute to use?**

Seems Straightforward

- What happens when we have more than one attribute?
- In the example before, we only had to consider annual income
- But what would happen if we had multiple attributes, like 5 or 10 or 100?
- **How do we decide which attribute to use?**

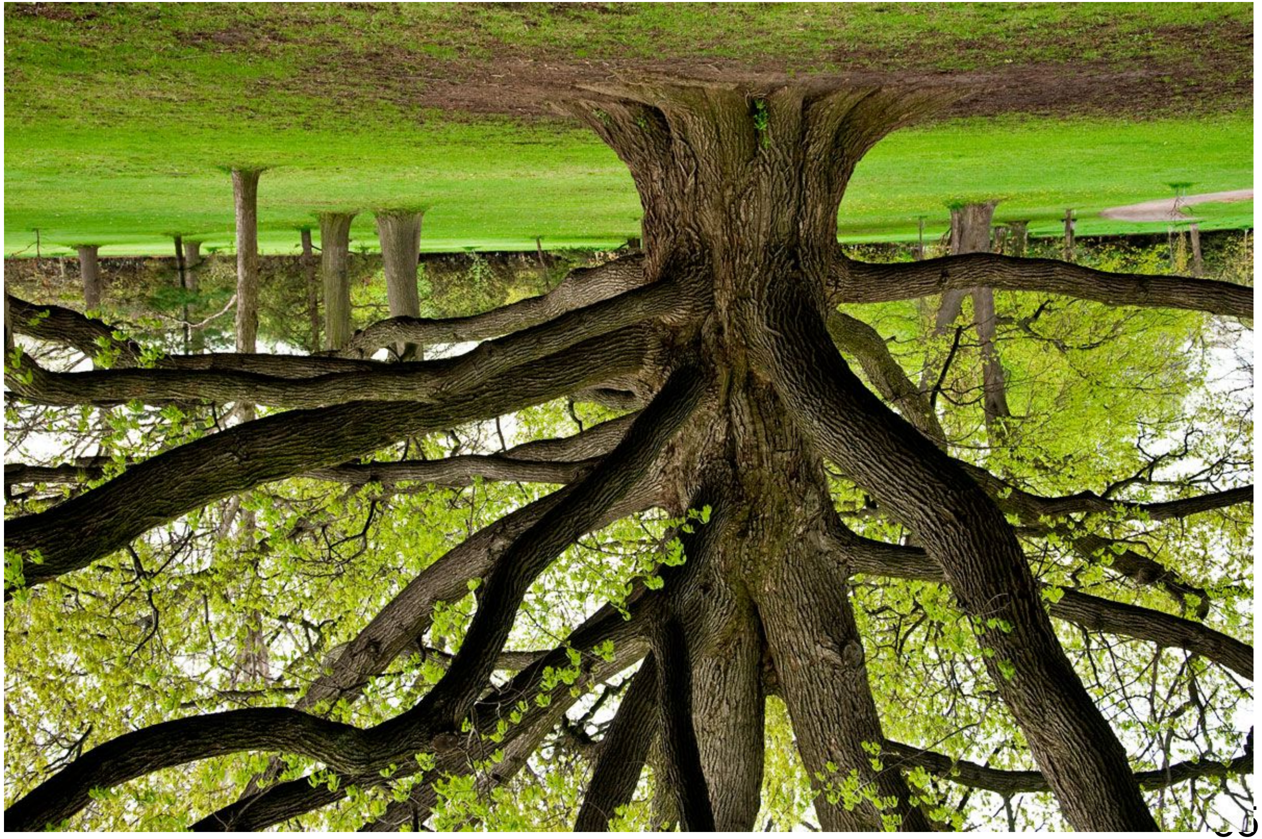
Decision Trees!

Trees & Decisions Trees

Regular Trees

Trees in Computer Science

Regular Trees

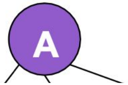


Trees in Computer Science

- A Decision Tree is a way for a computer to make decisions based on a series of questions.

Trees in Computer Science

- A Decision Tree is a way for a computer to make decisions based on a series of questions.



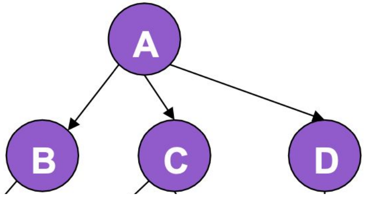
A **tree** is a **collection of nodes** such that

- One node is the designated ***root***.

A is the ***root***

Trees in Computer Science

- A Decision Tree is a way for a computer to make decisions based on a series of questions.



B, C and D are A's children

A **tree** is a **collection of nodes** such that

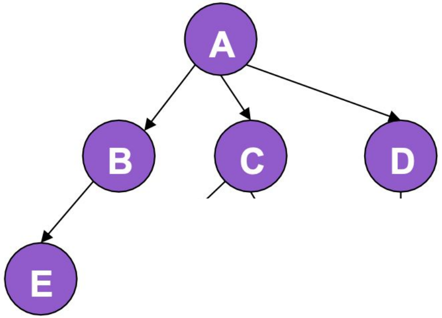
- One node is the designated ***root***.
- A node can have zero or more *children*;

Trees in Computer Science

- A Decision Tree is a way for a computer to make decisions based on a series of questions.

A **tree** is a **collection of nodes** such that

- One node is the designated ***root***.
- A node can have zero or more *children*;
- a node with zero children is a *leaf*.



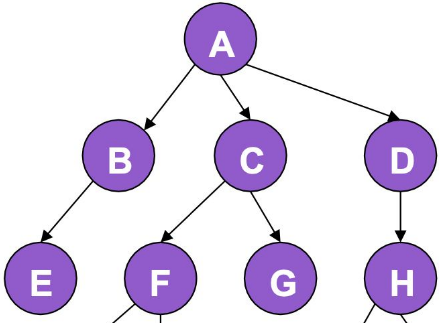
E is a leaf

Trees in Computer Science

- A Decision Tree is a way for a computer to make decisions based on a series of questions.

A **tree** is a **collection of nodes** such that

- One node is the designated ***root***.
- A node can have zero or more *children*;
- a node with zero children is a *leaf*.
- All non-root nodes have a single *parent*.



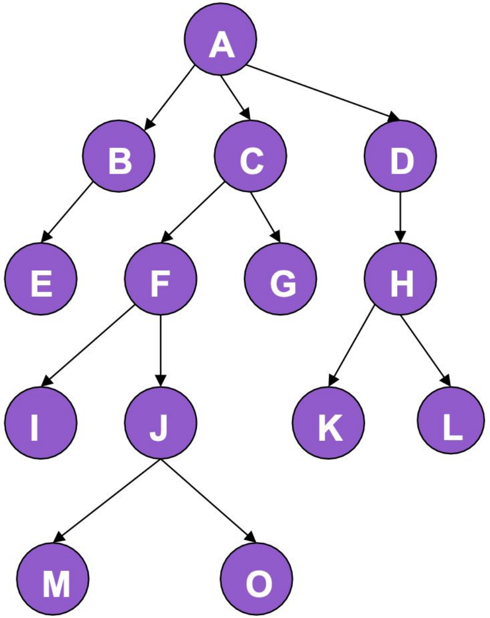
C is a parent to F + G

Trees in Computer Science

- A Decision Tree is a way for a computer to make decisions based on a series of questions.

A **tree** is a **collection of nodes** such that

- One node is the designated ***root***.
- A node can have zero or more *children*;
- a node with zero children is a *leaf*.
- All non-root nodes have a single *parent*.
- *Edges* denote parent-child relationships.
    - Example: The arrows between F → I

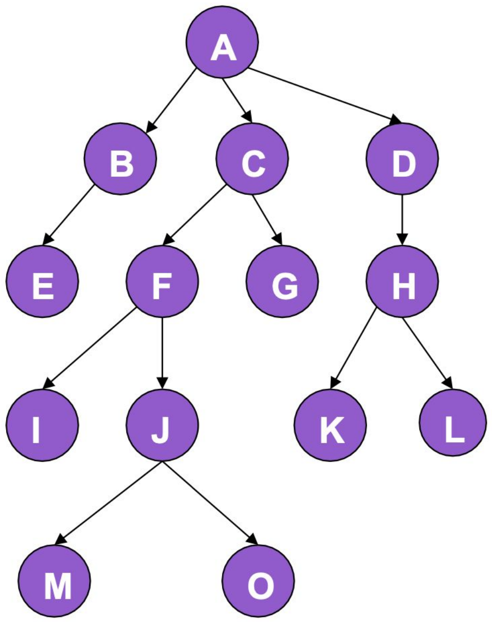


Trees in Computer Science

- A Decision Tree is a way for a computer to make decisions based on a series of questions.

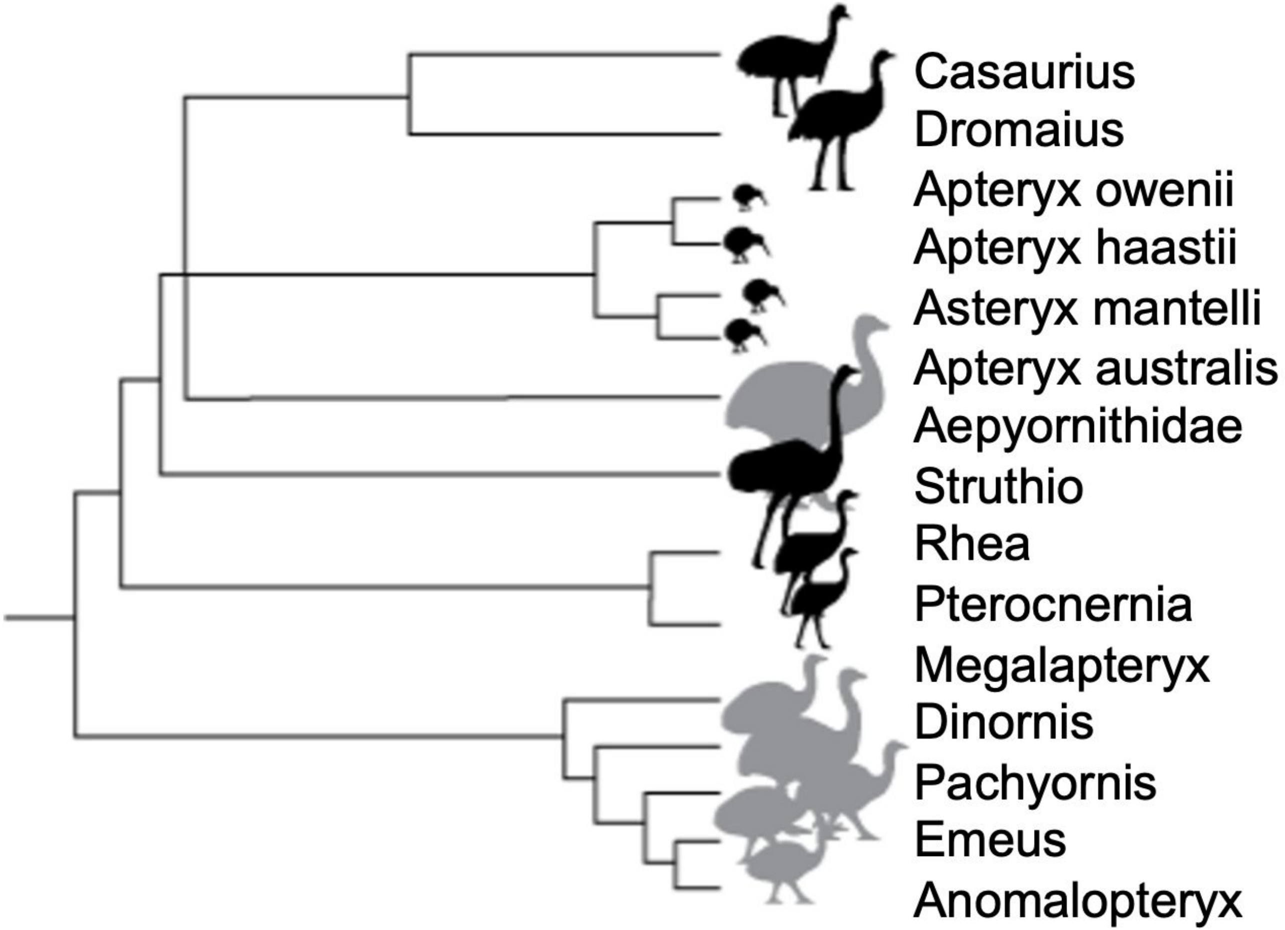
A **tree** is a **collection of nodes** such that

- One node is the designated ***root***.
- A node can have zero or more *children*;
- a node with zero children is a *leaf*.
- All non-root nodes have a single *parent*.
- *Edges* denote parent-child relationships.
- Nodes and/or edges may be labeled by data.
  - Each node on this tree is labeled by a letter



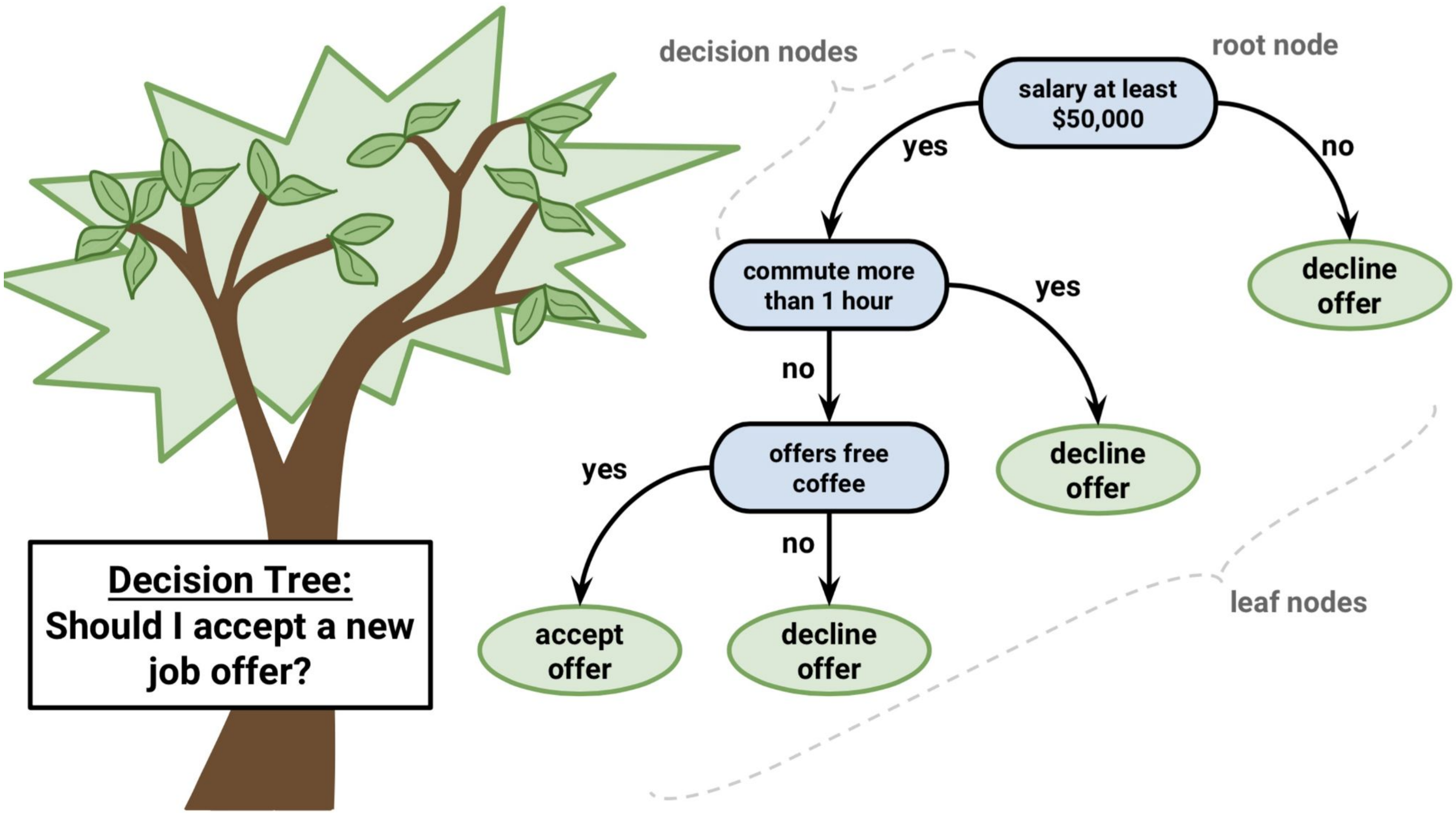
Non-CS Decisions Trees

Rooted trees in CS often (but not always) drawn with root on top



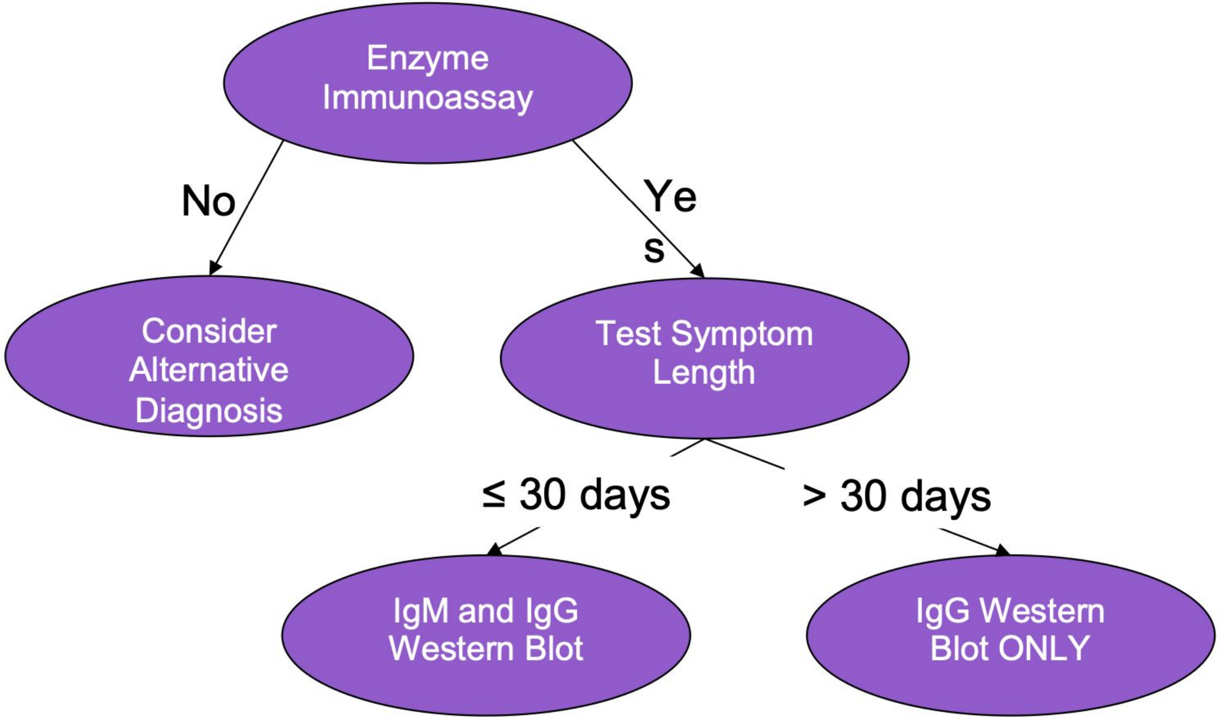
Decision trees

Trees whose node labels are attributes, edge labels are conditions



Decisions Trees in Medicine

Decision tree for Lyme Disease diagnosis



Decisions Trees in Business

GBR Graziadio Business Review
A Peer-Reviewed Journal of Relevant Information and Analysis

Home | Archives | About | Submissions | Blog

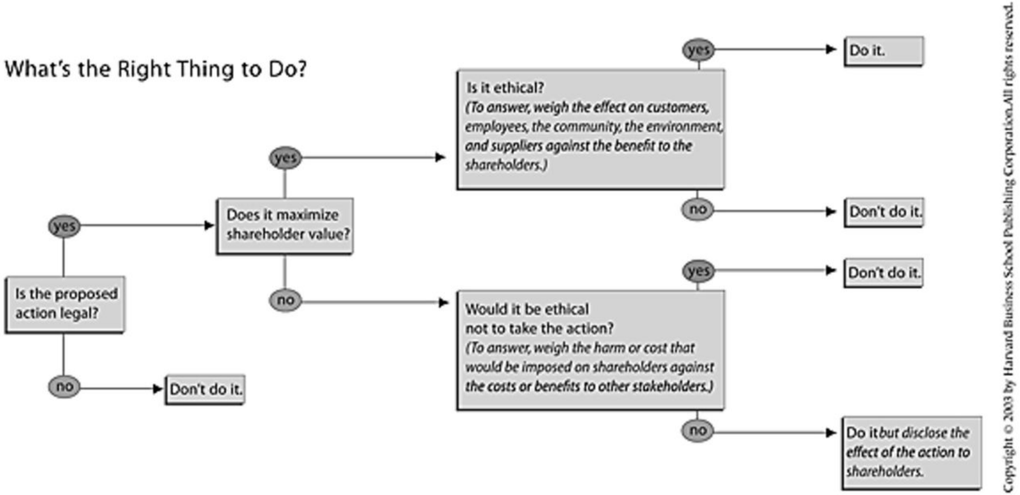
How Gerber Used a Decision Tree in Strategic Decision-Making

Possible outcomes explored in an in
Products Safety Commission.

By JAY BUCKLEY and THOMAS J. DUDLEY, DBA

1999 Volume 2 Issue 3

Decision trees can assist executives in making strat



Let's build a Decision Tree

Building Decision Trees

- Should you get an ice cream?
- You might start out with the following data

Weather	Wallet	Ice Cream?
Great	Empty	No
Nasty	Empty	No
Great	Full	Yes
Okay	Full	Yes
Nasty	Full	No

Building Decision Trees

- Should you get an ice cream?
- You might start out with the following data

Attributes

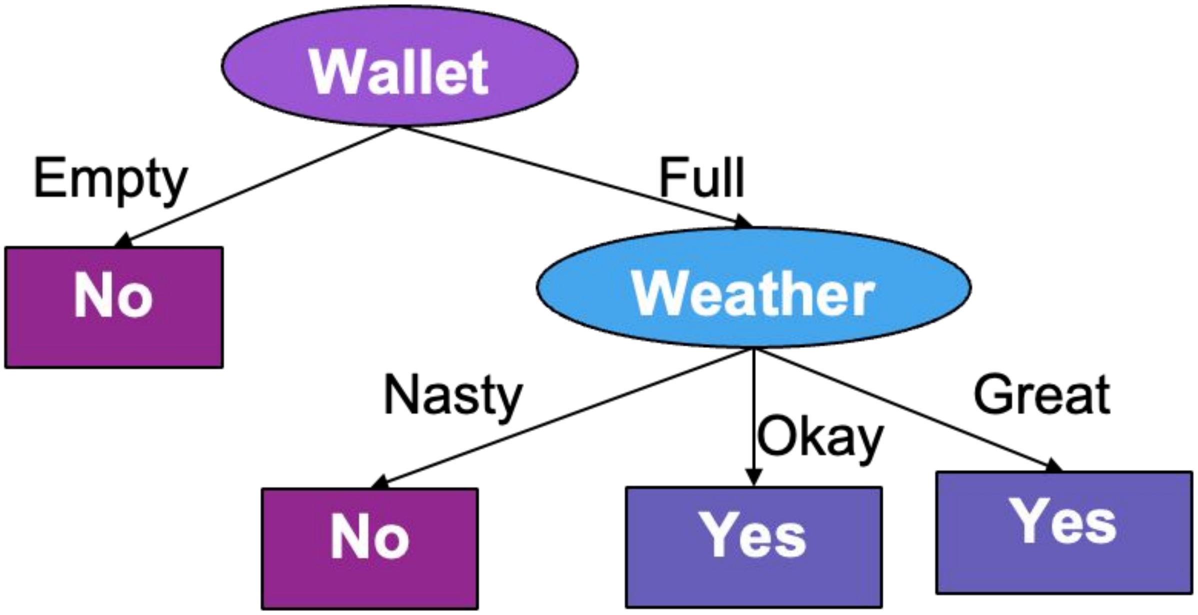
Conditions

Weather	Wallet	Ice Cream?
Great	Empty	No
Nasty	Empty	No
Great	Full	Yes
Okay	Full	Yes
Nasty	Full	No

Ice Cream Decision Tree

Should you get an ice cream?

Weather	Wallet	Ice Cream?
Great	Empty	No
Nasty	Empty	No
Great	Full	Yes
Okay	Full	Yes
Nasty	Full	No



Another Example

Soccer League:

Do we cancel the game?

Soccer League Data

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

Soccer League: Cancel Game?

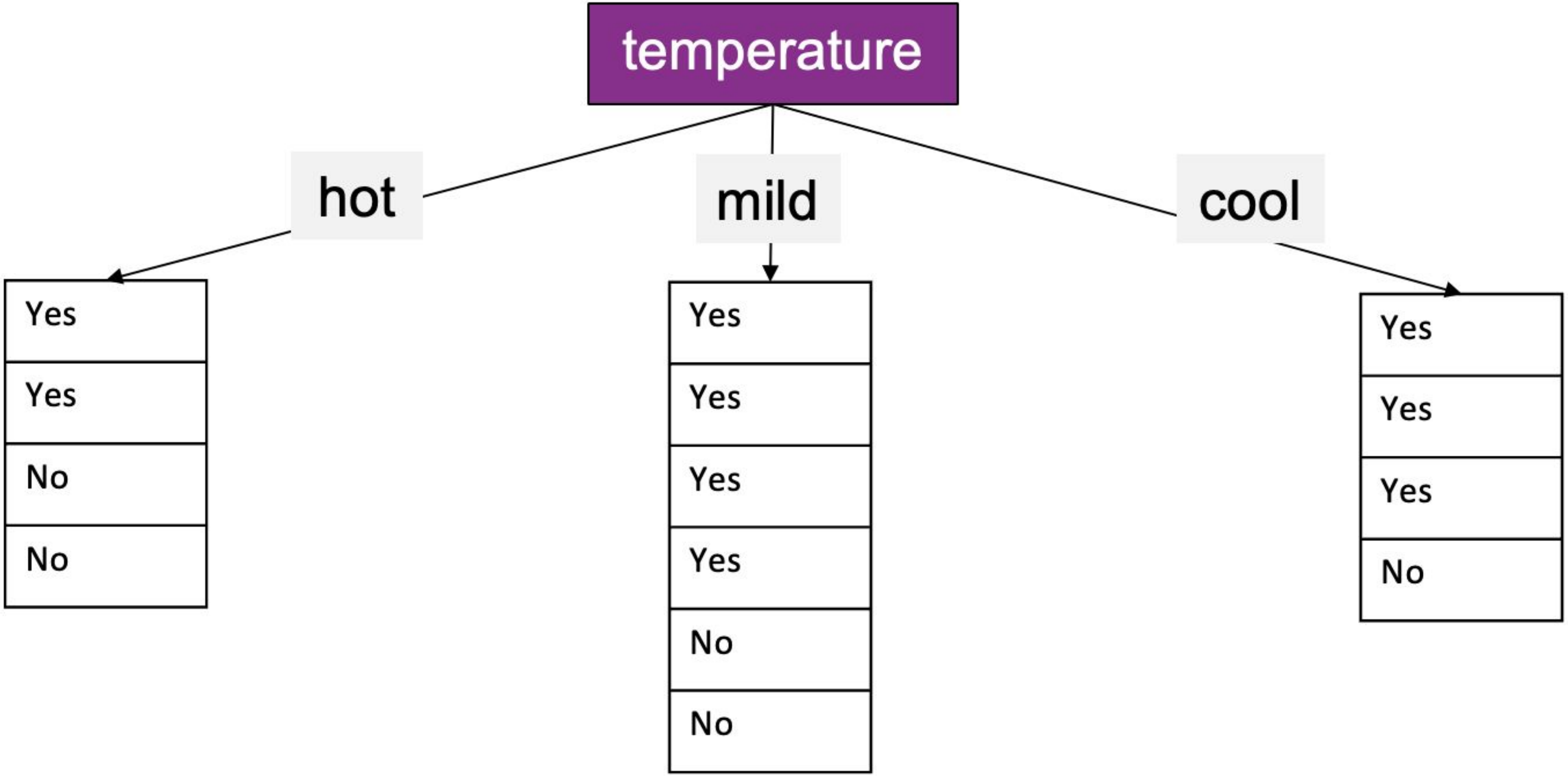
- Build a decision tree to help officials decide
- Assume that decisions are the same given the same information
- The leaf nodes should be whether or not to play
- The non-leaf nodes should be attributes (e.g., Outlook, Windy)
- The edges should be conditions (e.g., sunny, hot, normal)

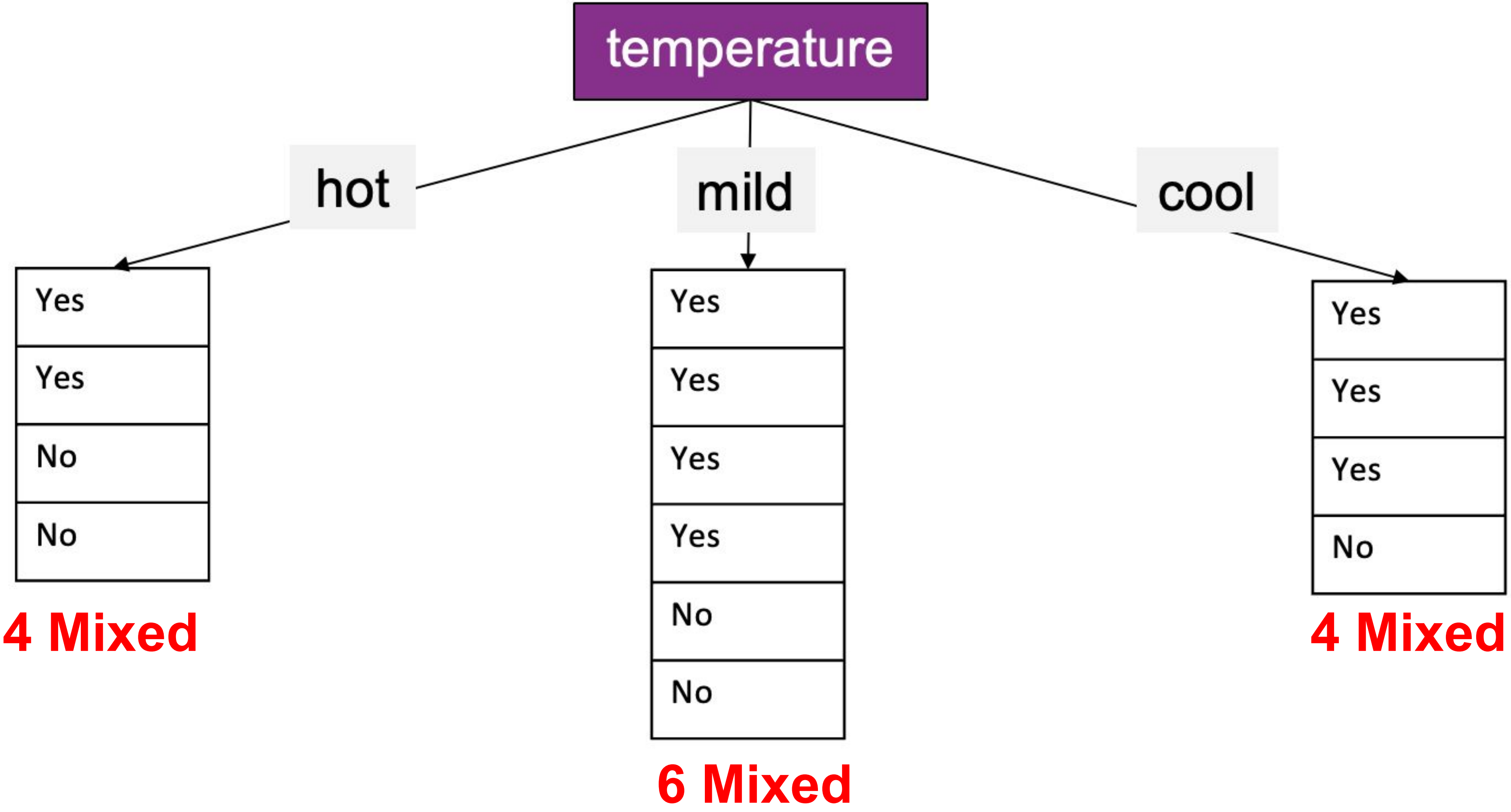
Want to have as few mixed “Yes” and “No” answers together in groups as possible.

At the start we have 14 mixed Yes’s/No’s

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

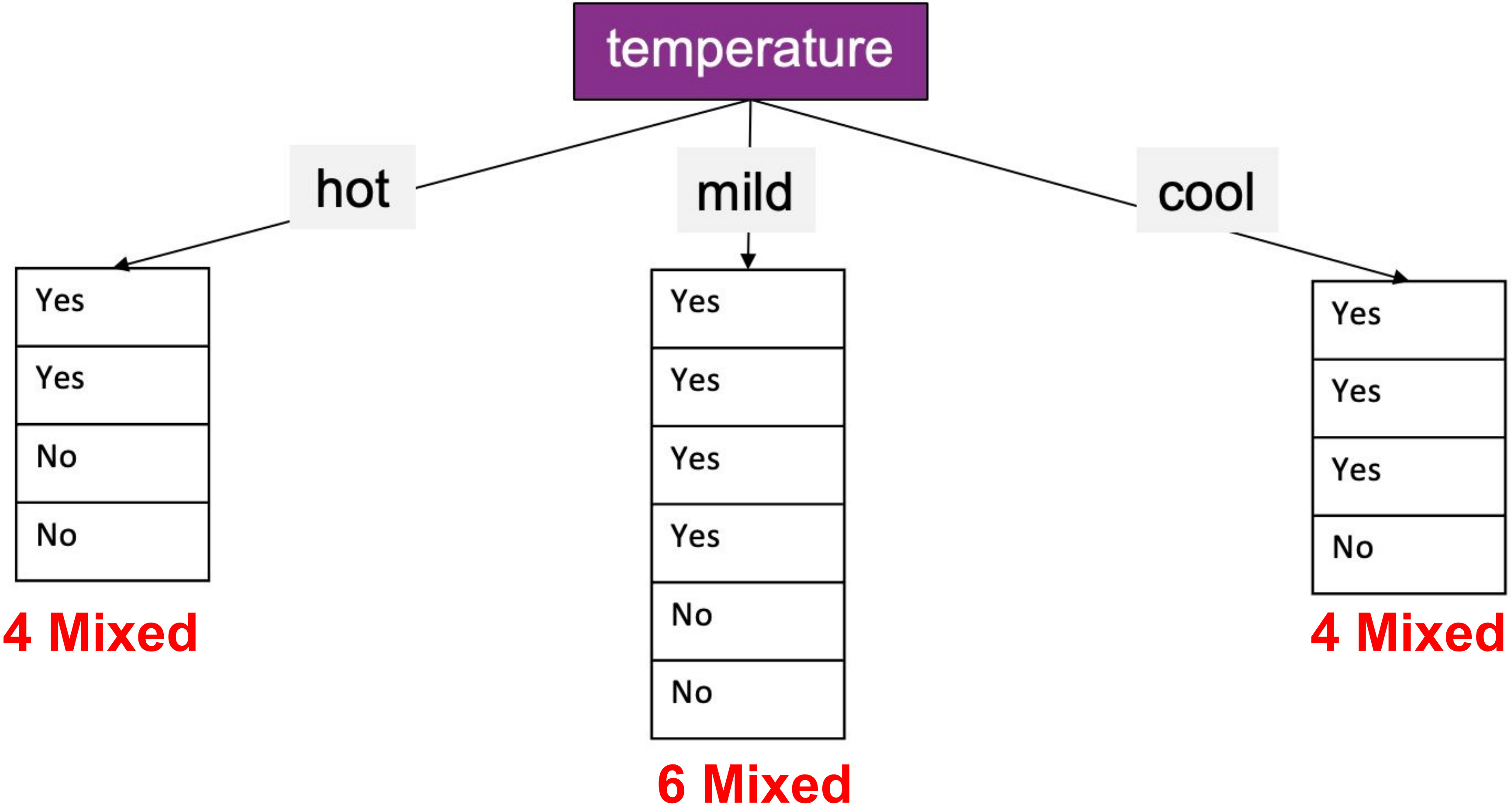
What happens if we split data on Temperature?


temperature
hot
mild
cool
Yes
Yes
No
No
Yes
Yes
Yes
Yes
No
No
Yes
Yes
Yes
No

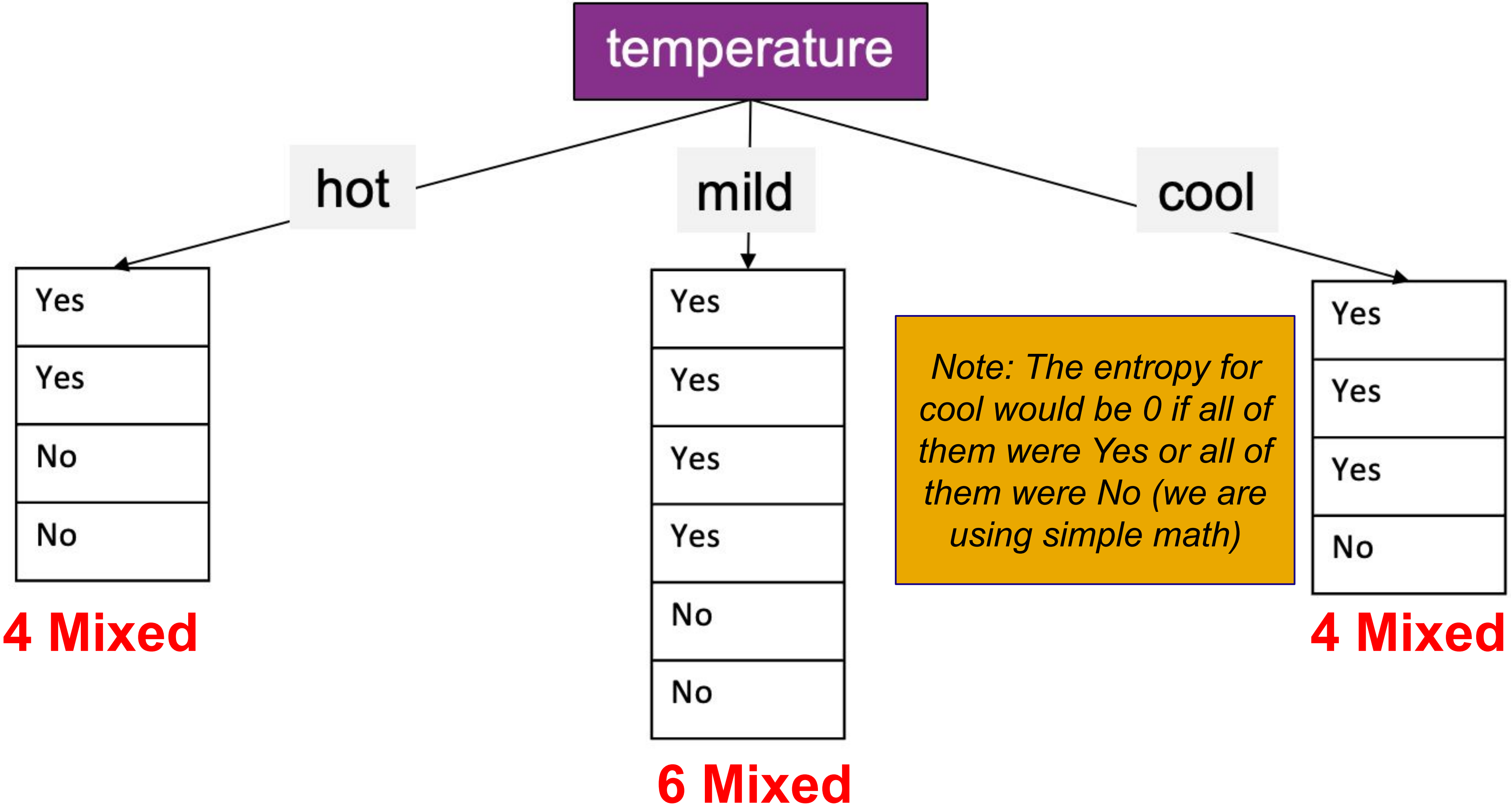

temperature
hot
mild
cool
Yes
Yes
No
No
Yes
Yes
Yes
Yes
No
No
Yes
Yes
Yes
No
4 Mixed
6 Mixed
4 Mixed

What is the uncertainty (entropy) in our data?

Overall entropy = 4 + 4 + 6 = 14



Overall entropy = 4 + 4 + 6 = 14



Note: The entropy for cool would be 0 if all of them were Yes or all of them were No (we are using simple math)

65

In-class Activity

[Groups of 2-3]

What's the entropy if you split on Outlook?

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

What's the entropy if you split on Windy?

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

What's the entropy if you split on Humidity?

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

[Placeholder for Clicker]

Recap

What is the best attribute to split on?

- Entropy if we split on Temperature = 14
- Entropy if we split on Outlook = 10
- Entropy if we split on Windy = 14
- Entropy if we split on Humidity = 14

Why? It does the best job of **reducing** entropy

Wrap up