

CSE419 – Artificial Intelligence and Machine Learning 2018

PhD Furkan Gözükar, Toros University

https://github.com/FurkanGozukara/CSE419_2018

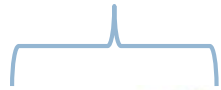
Lecture 3

Decision Trees

Based on Asst. Prof. Dr. David Kauchak (Pomona College) Lecture Slides

Representing examples

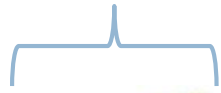
examples



What is an example?
How is it represented?

Features

examples



features

$f_1, f_2, f_3, \dots, f_n$

$f_1, f_2, f_3, \dots, f_n$

$f_1, f_2, f_3, \dots, f_n$

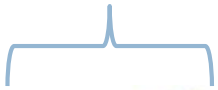
$f_1, f_2, f_3, \dots, f_n$

How our algorithms actually “view” the data

Features are the questions we can ask about the examples

Features

examples



features

red, round, leaf, 3oz, ...

green, round, no leaf, 4oz, ...

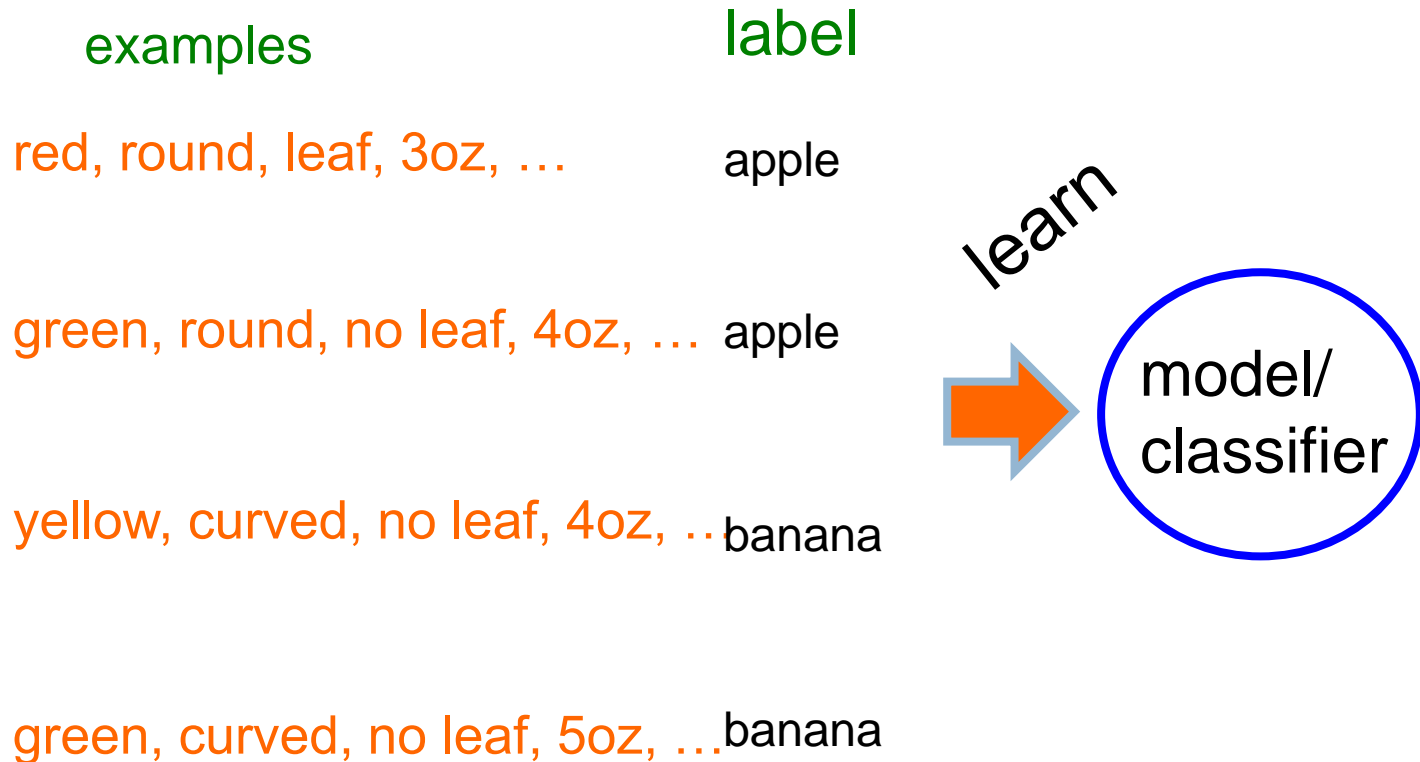
yellow, curved, no leaf, 4oz, ...

green, curved, no leaf, 5oz, ...

How our algorithms actually “view” the data

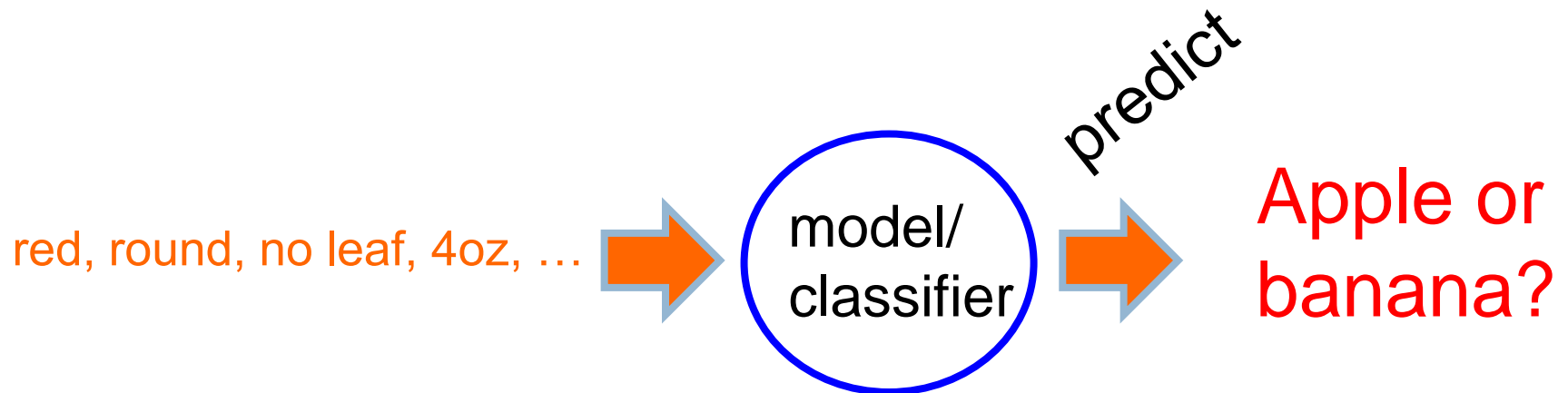
Features are the questions we can ask about the examples

Classification revisited



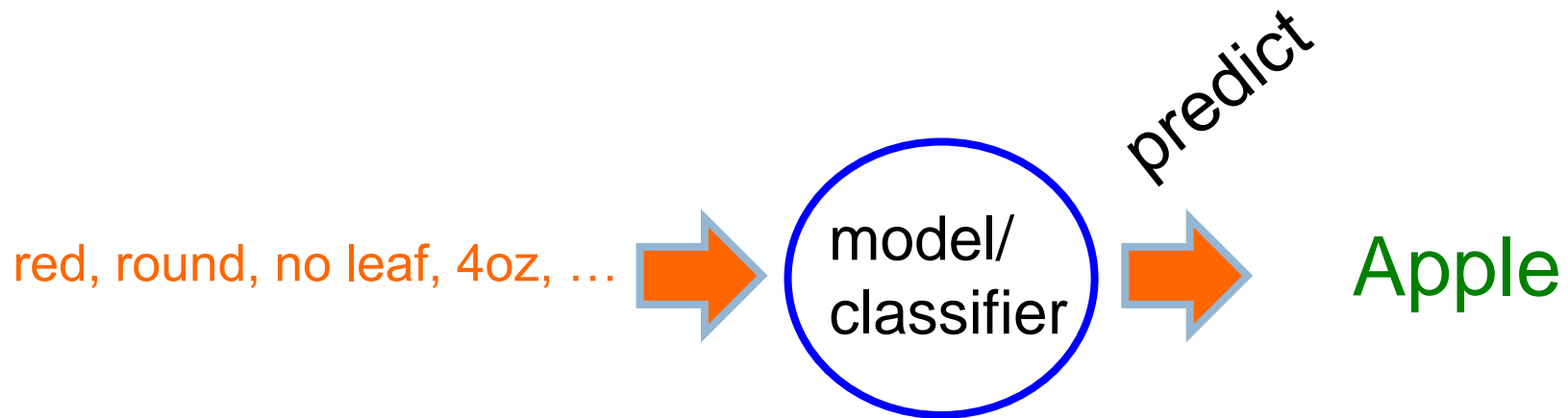
During learning/training/induction, learn a model of what distinguishes apples and bananas *based on the features*

Classification revisited



The model can then classify a new example *based on the features*

Classification revisited



Why?

The model can then classify a new example *based on the features*

Classification revisited

Training data

examples

label

red, round, leaf, 3oz, ...

apple

green, round, no leaf, 4oz, ...

apple

yellow, curved, no leaf, 4oz, ...

banana

green, curved, no leaf, 5oz, ...

banana

Test set

red, round, no leaf, 4oz, ...?

Classification revisited

Training data

examples

red, round, leaf, 3oz, ...

label

apple

green, round, no leaf, 4oz, ...

apple

yellow, curved, no leaf, 4oz, ...

banana

green, curved, no leaf, 5oz, ...

banana

Test set

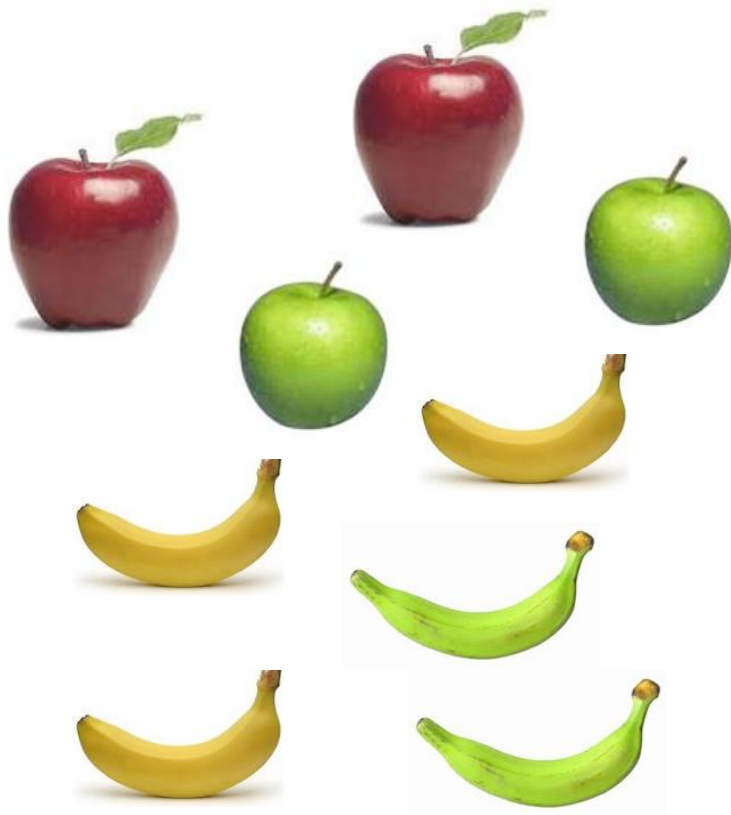
red, round, no leaf, 4oz, ...?

Learning is about
generalizing from the
training data

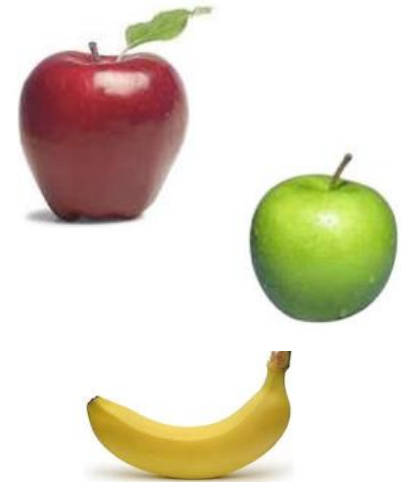
What does this assume
about the training and test
set?

Past predicts future

Training data

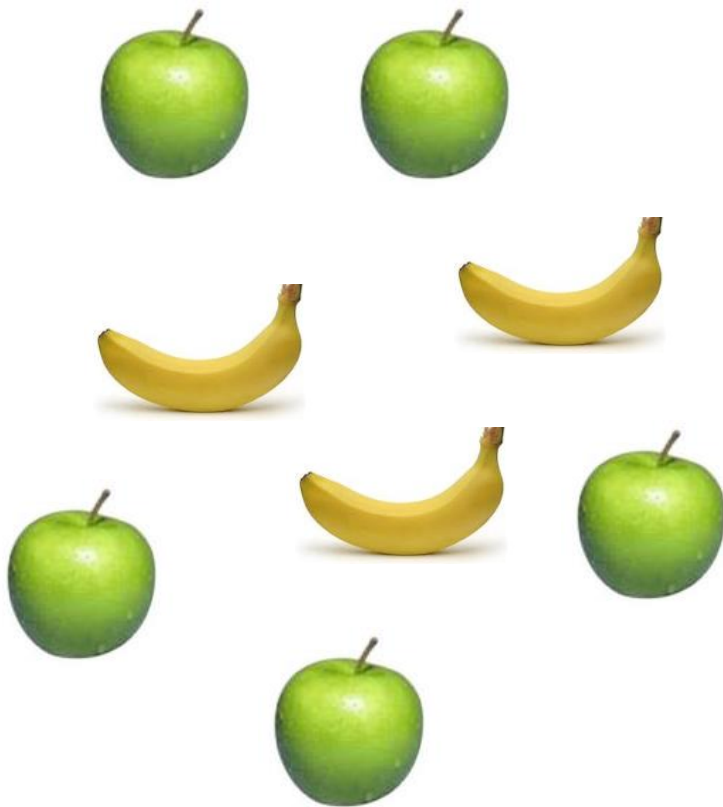


Test set

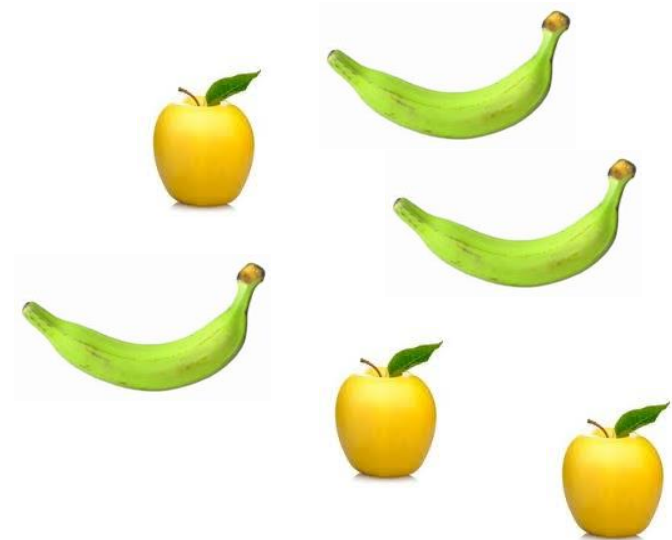


Past predicts future

Training data



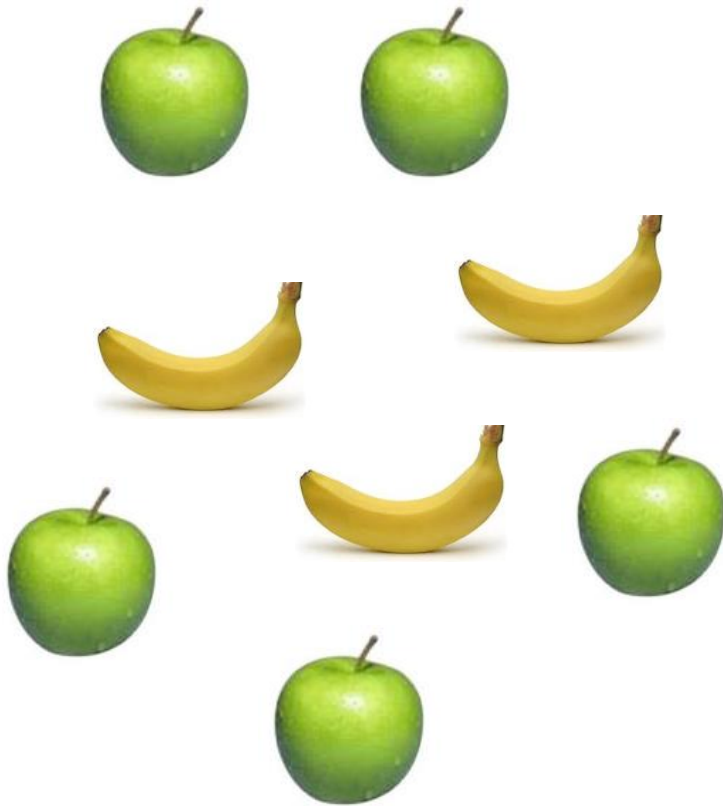
Test set



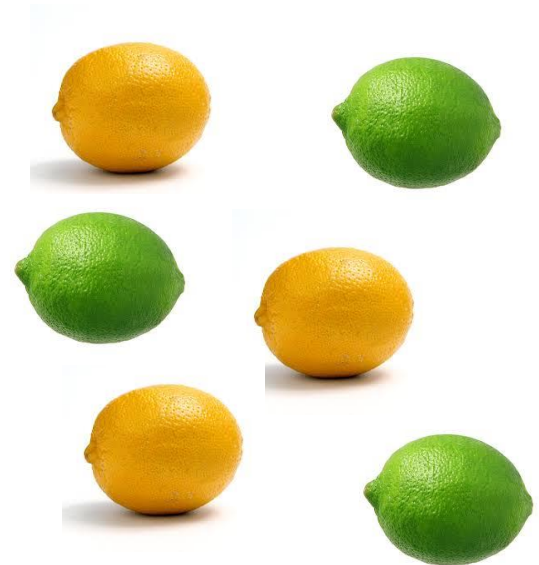
Not always the case, but we'll often assume it is!

Past predicts future

Training data



Test set



Not always the case, but we'll often assume it is!

More technically...

We are going to use the *probabilistic model* of learning

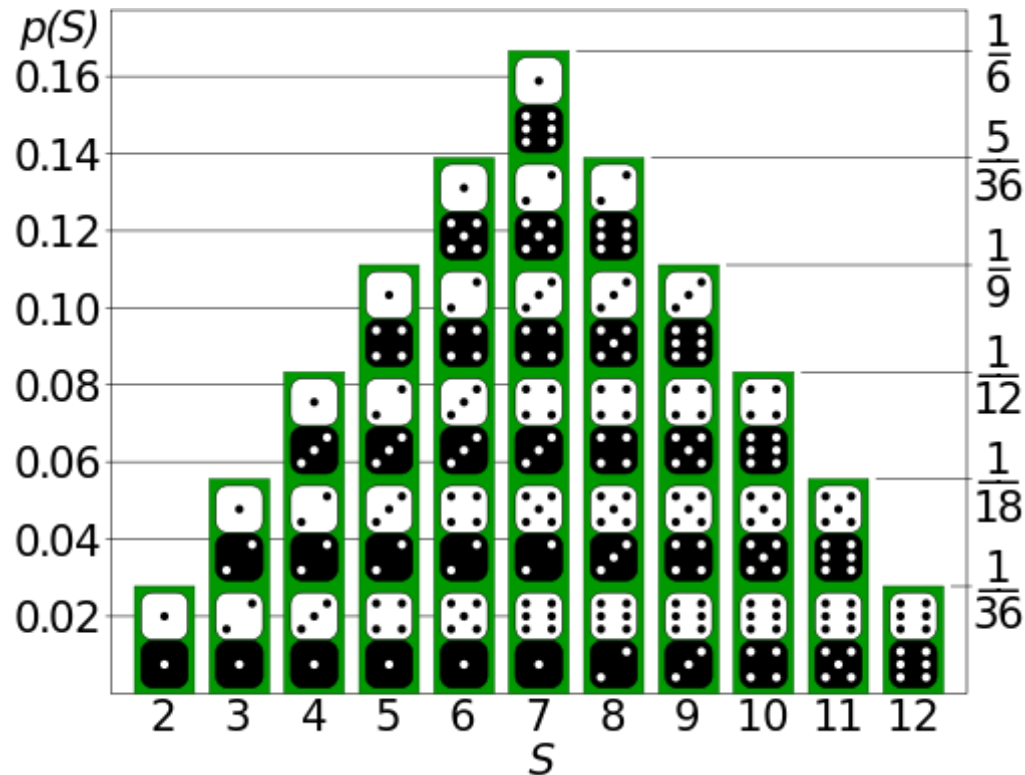
There is some probability distribution over example/label pairs called the *data generating distribution*

Both the training data **and** the test set are generated based on this distribution

What is a probability distribution?

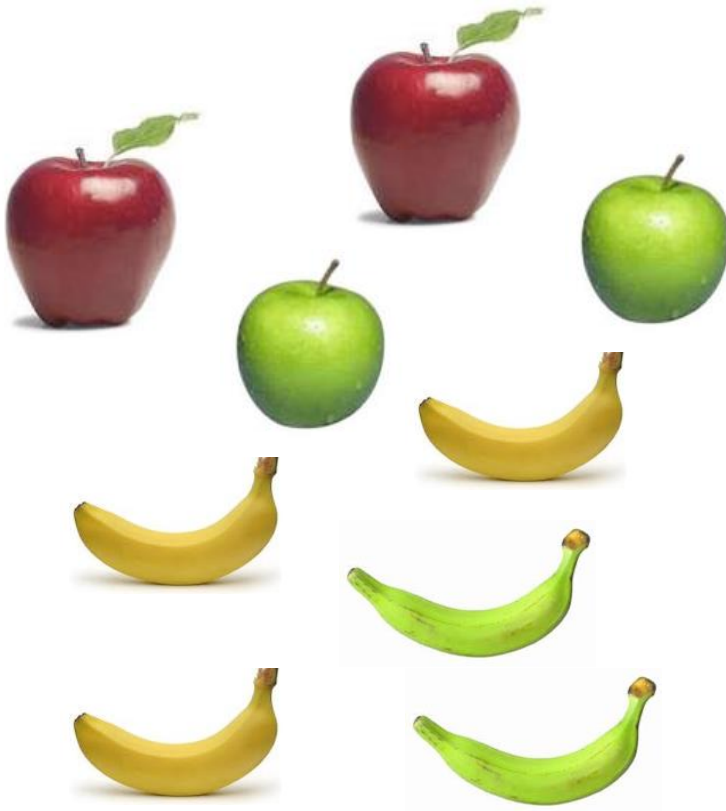
Probability distribution

Describes how likely (i.e. probable) certain events are



Probability distribution

Training data



High probability

round apples

curved bananas

apples with leaves

...

Low probability

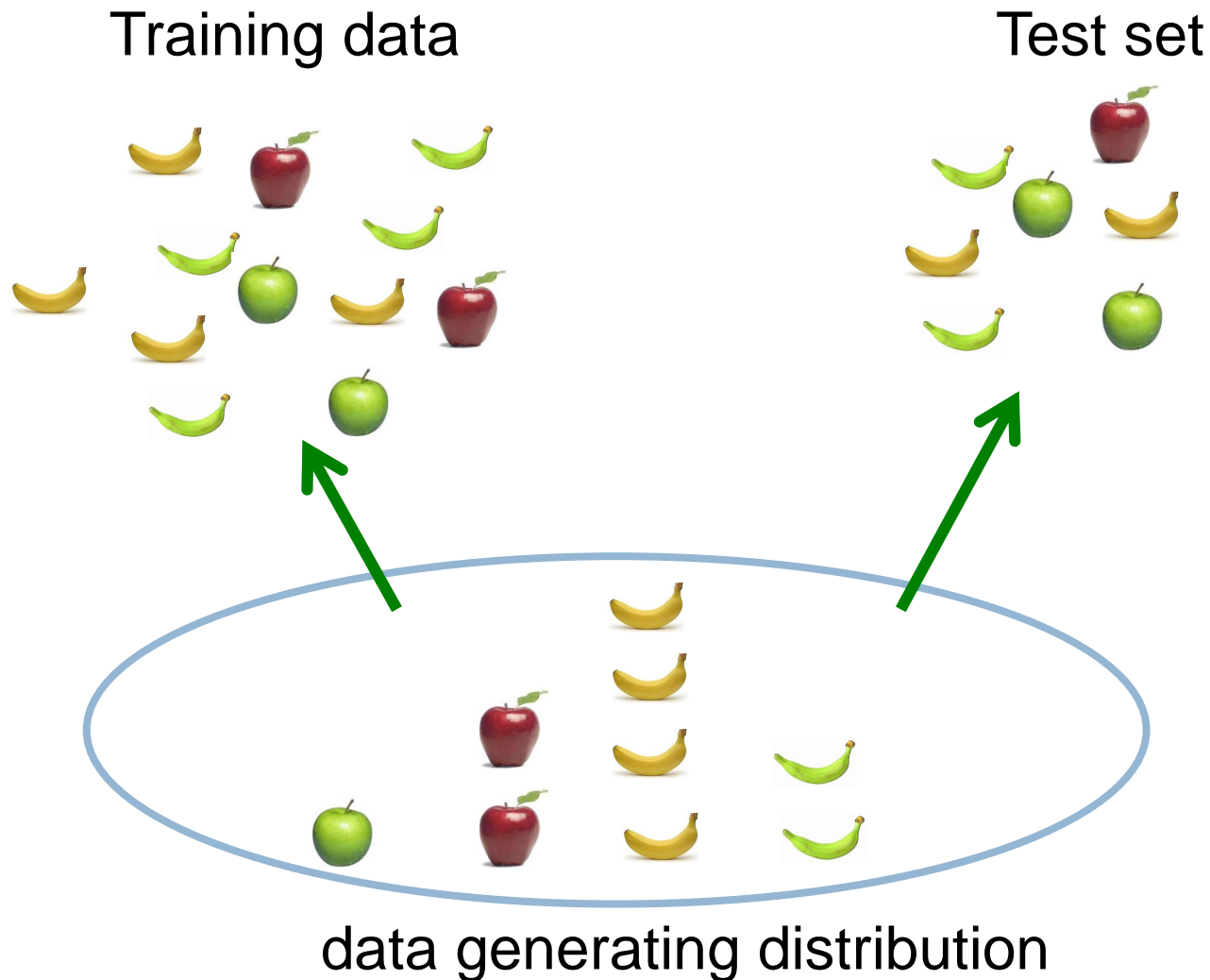
curved apples

red bananas

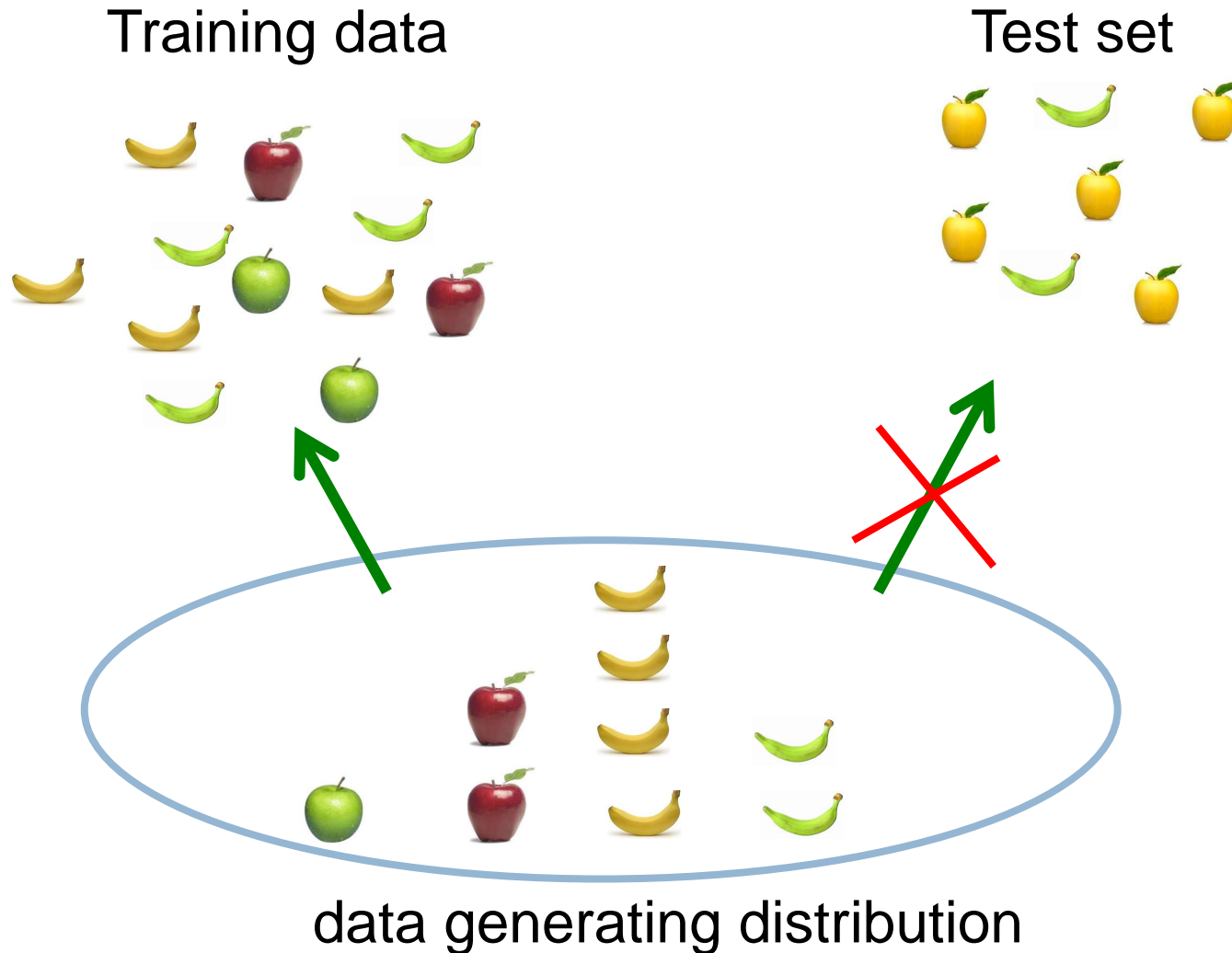
yellow apples

...

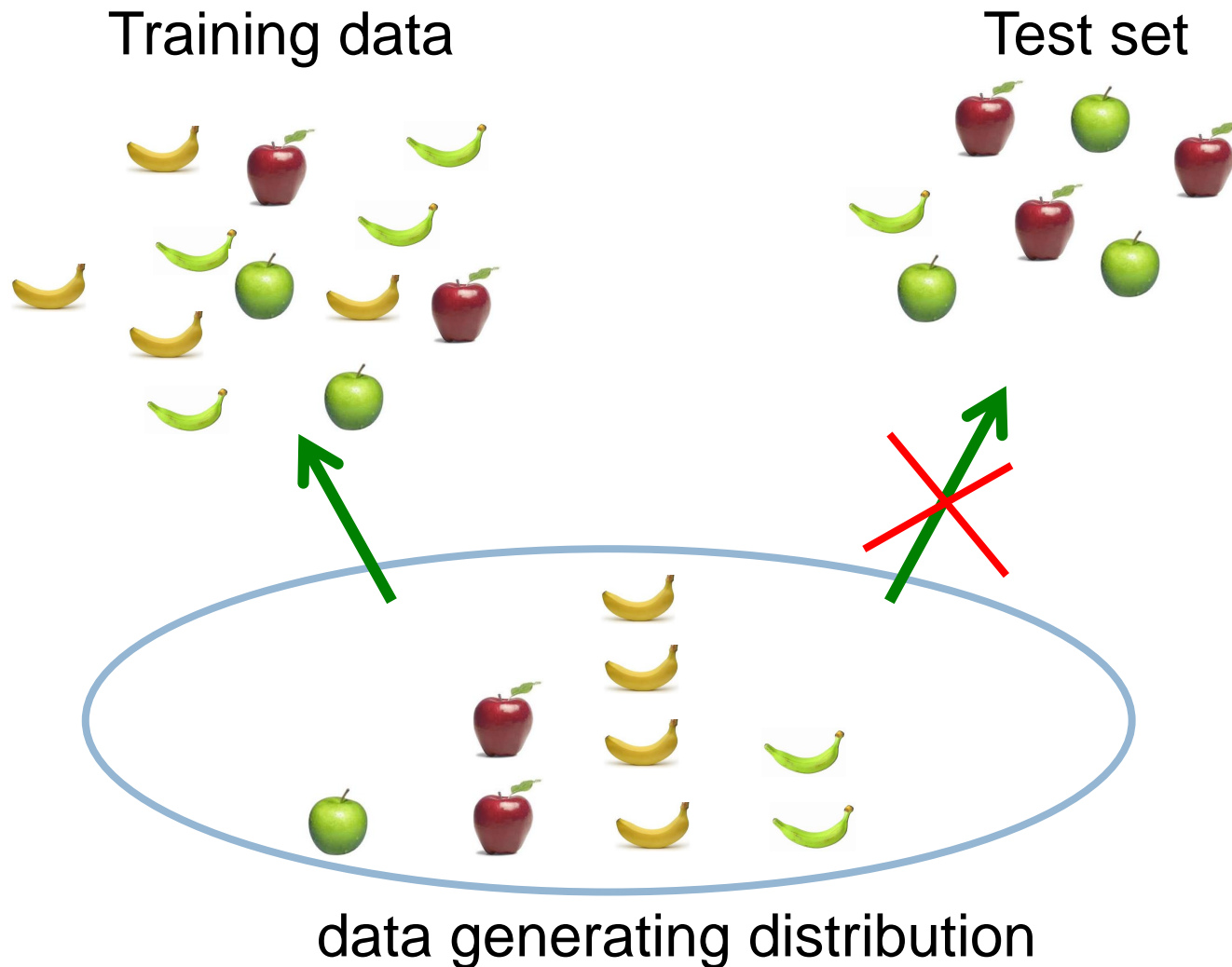
data generating distribution



data generating distribution



data generating distribution



To ride or not to ride, that is the question...

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

Build a decision tree

Recursive approach

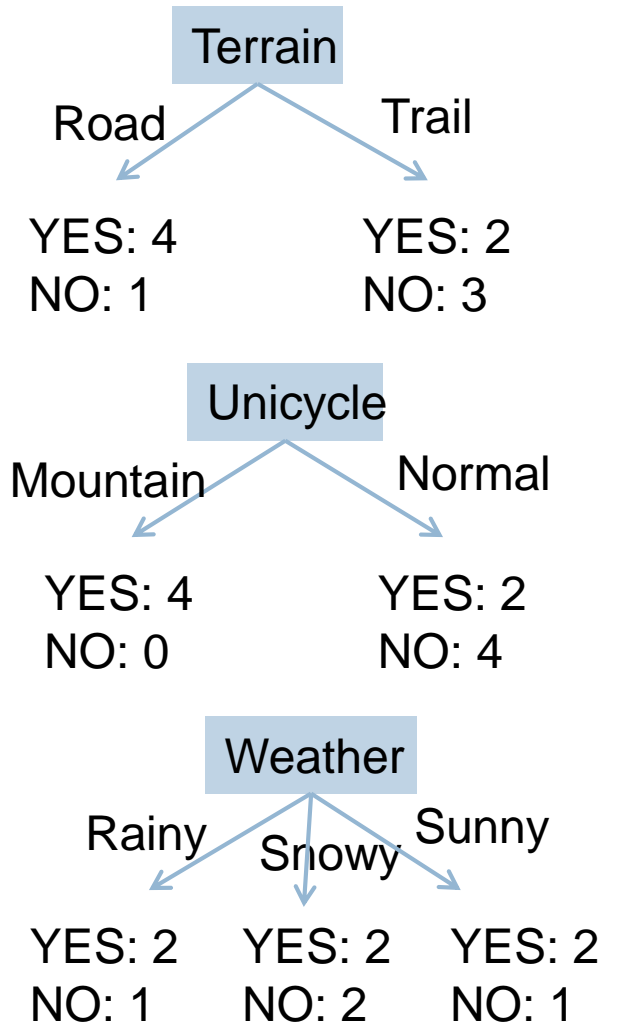
Base case: If all data belong to the same class, create a leaf node with that label

Otherwise:

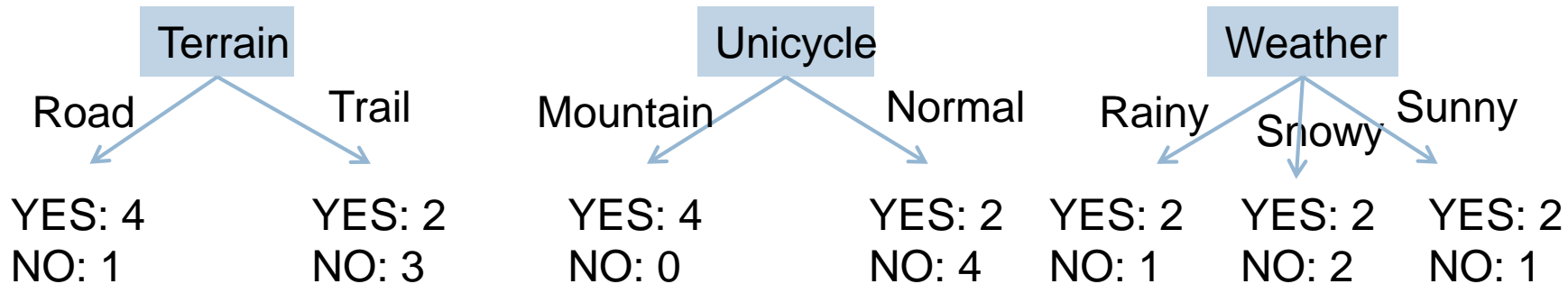
- calculate the “score” for each feature if we used it to split the data
- pick the feature with the highest score, partition the data based on that data value and call recursively

Partitioning the data

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES



Partitioning the data

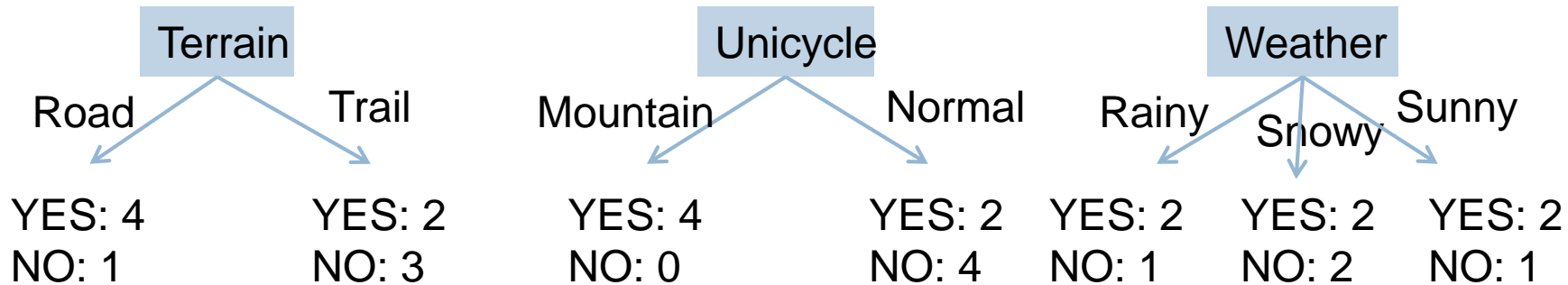


calculate the “**score**” for each feature if we used it to split the data

What score should we use?

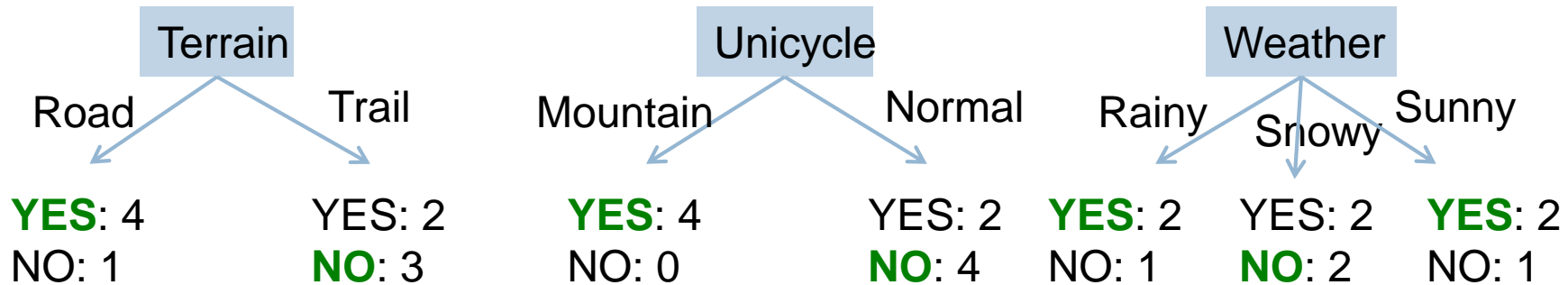
If we just stopped here, which tree would be best? How could we make these into decision trees?

Decision trees

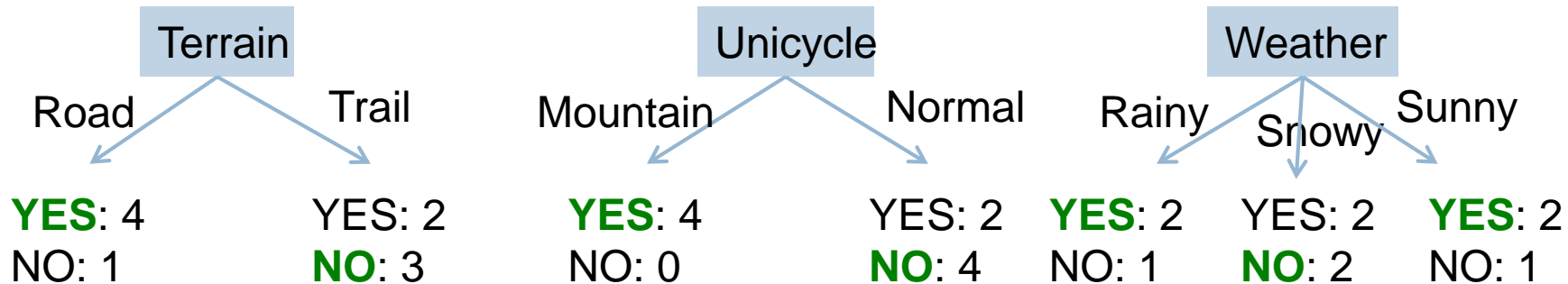


How could we make these into decision trees?

Decision trees



Decision trees

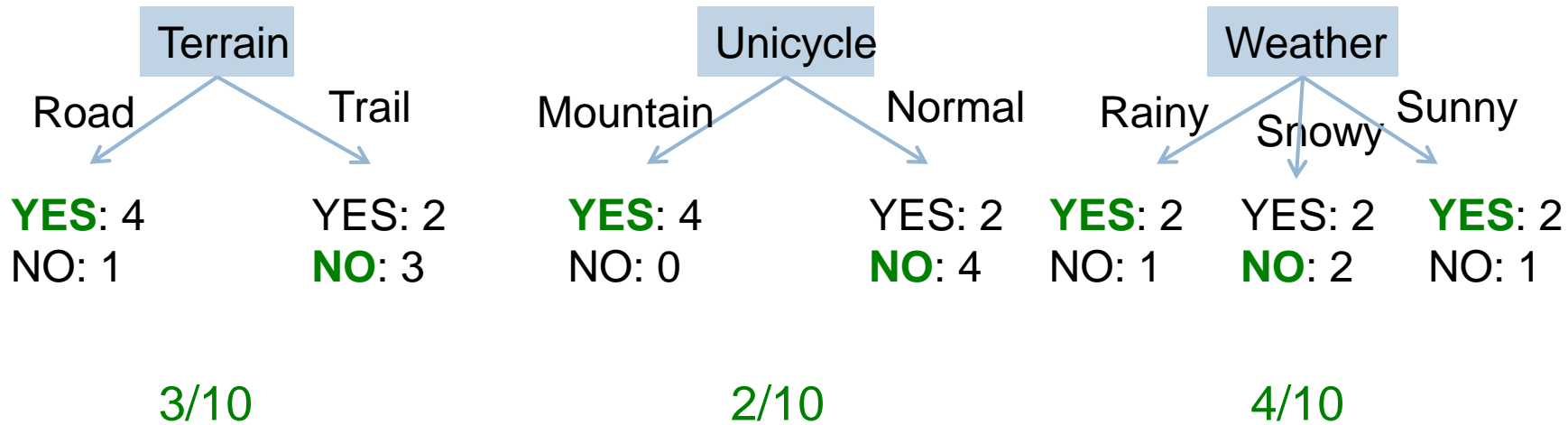


Training error: the average error over the training set

For classification, the most common “error” is the number of mistakes

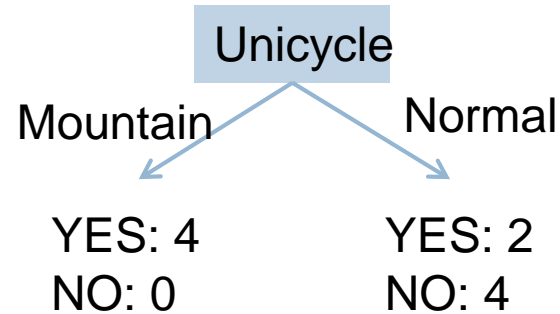
Training error for each of these?

Decision trees



Training error: the average error over the training set

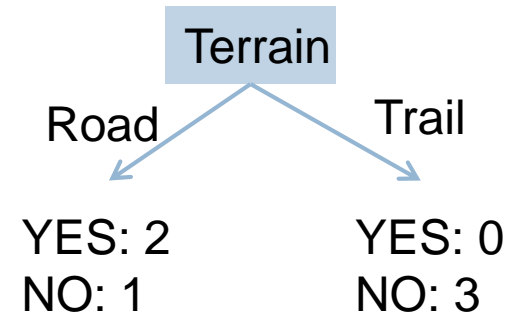
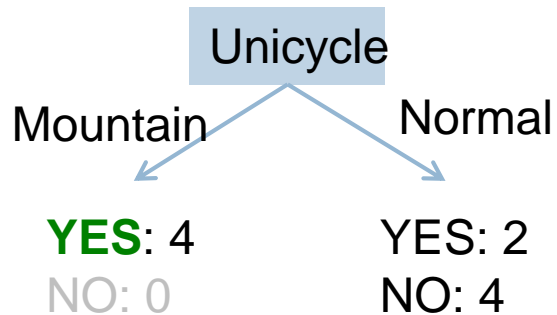
Recurse



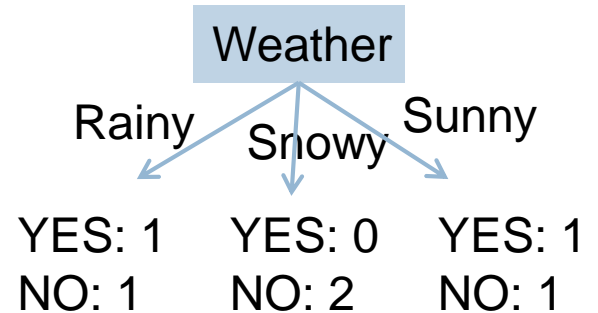
Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Mountain	Snowy	YES

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO

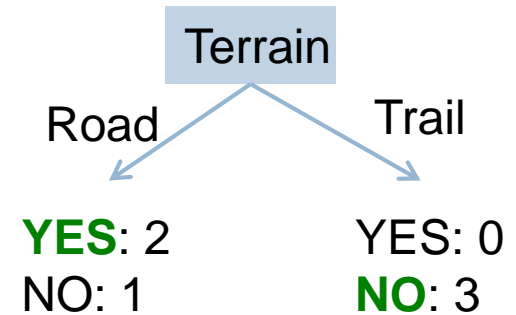
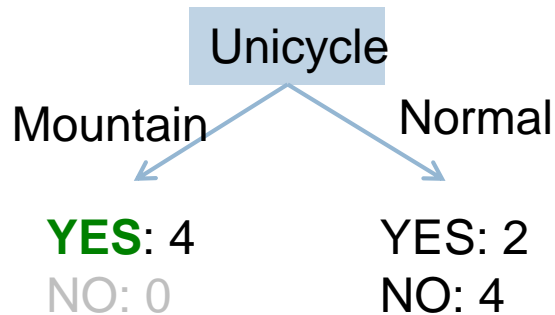
Recurse



Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO

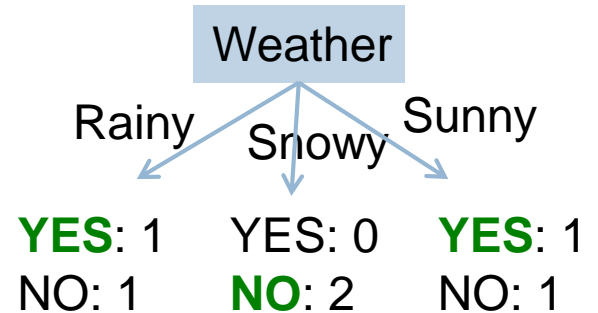


Recurse



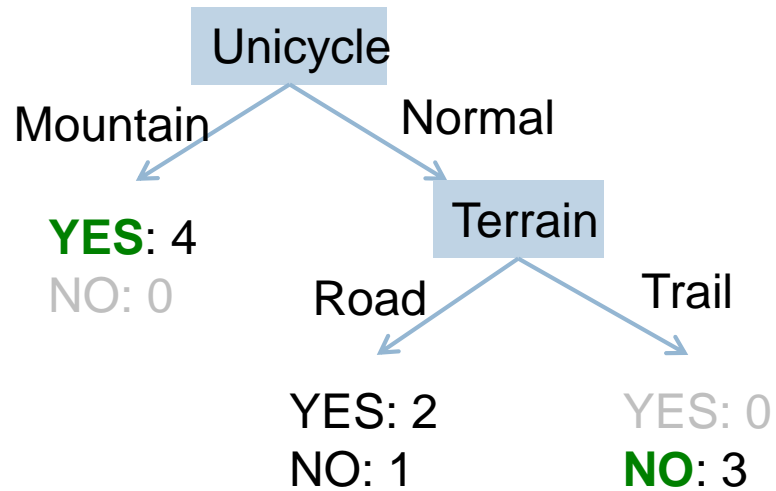
1/6

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO



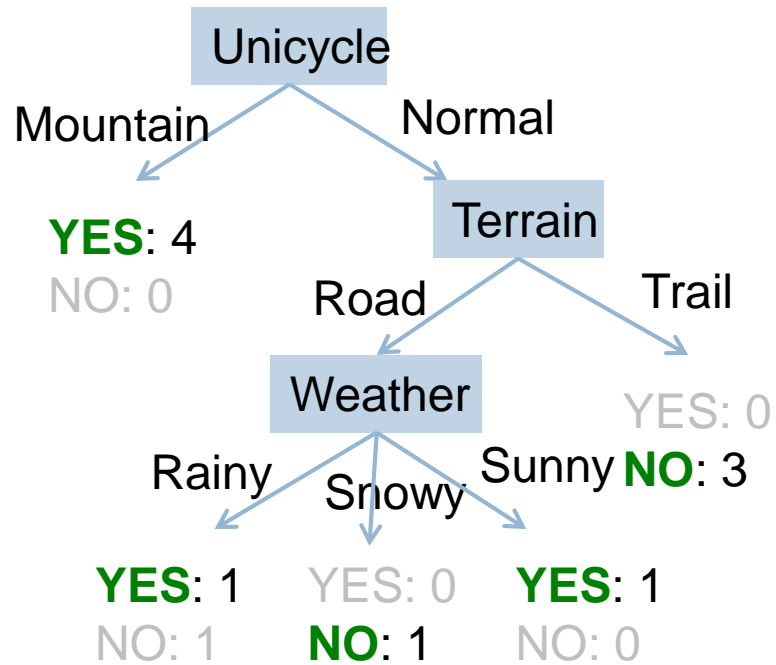
2/6

Recurse



Terrain	Unicycle-type	Weather	Go-For-Ride?
Road	Normal	Sunny	YES
Road	Normal	Rainy	YES
Road	Normal	Snowy	NO

Recurse



Building decision trees

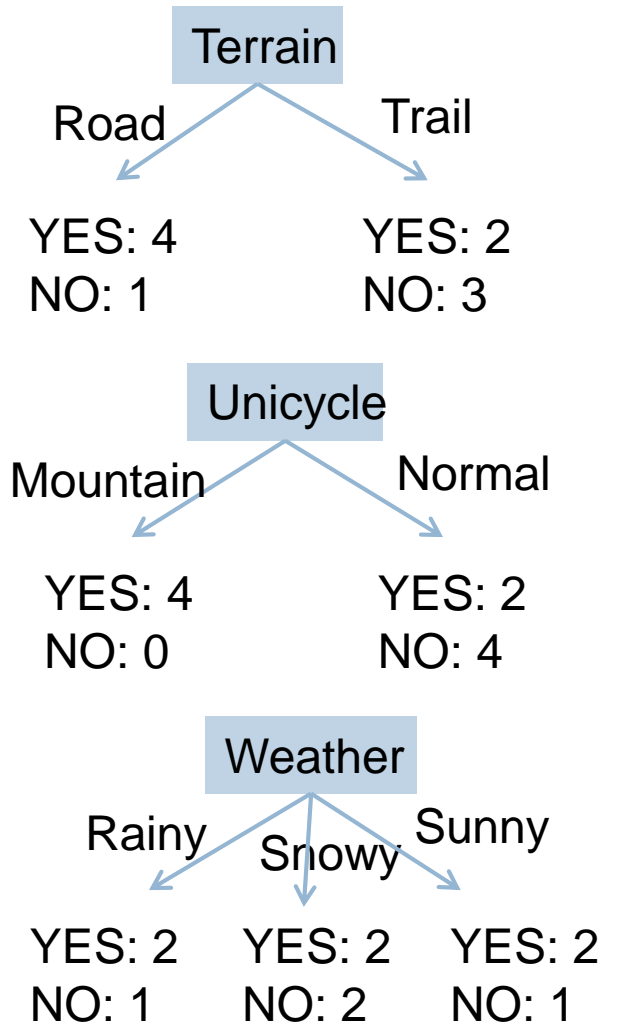
Base case: If all data belong to the same class, create a leaf node with that label

Otherwise:

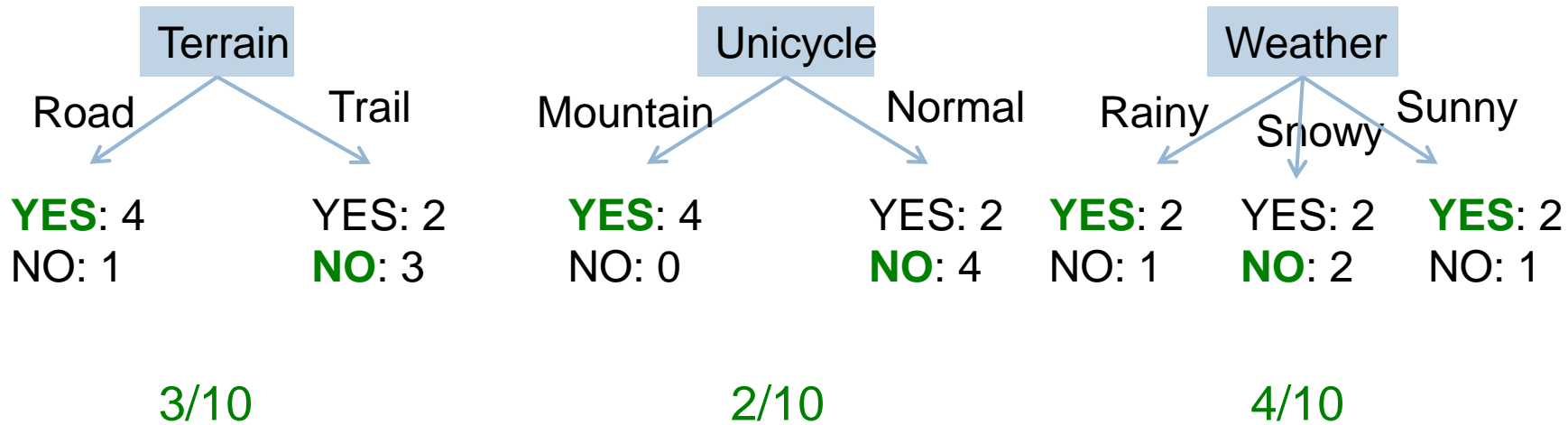
- calculate the “score” for each feature if we used it to split the data
- pick the feature with the highest score, partition the data based on that data value and call recursively

Partitioning the data

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

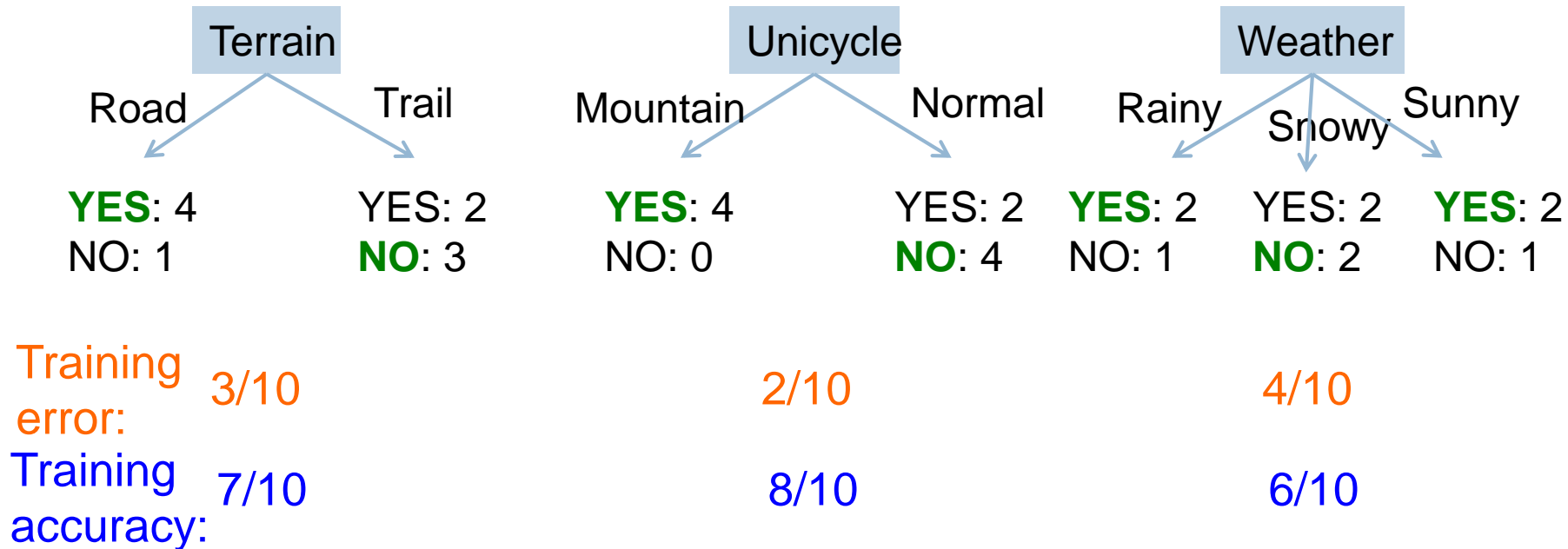


Decision trees



Training error: the average error over the training set

Training error vs. accuracy

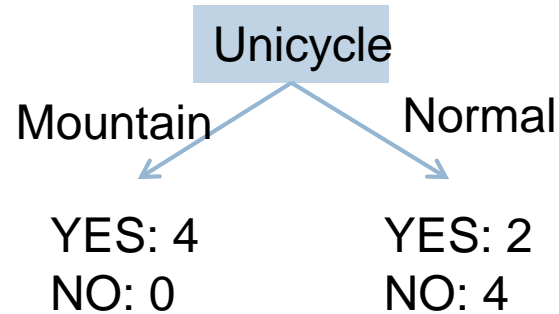


training error = 1-accuracy (and vice versa)

Training error: the average error over the training set

Training accuracy: the average percent correct over the training set

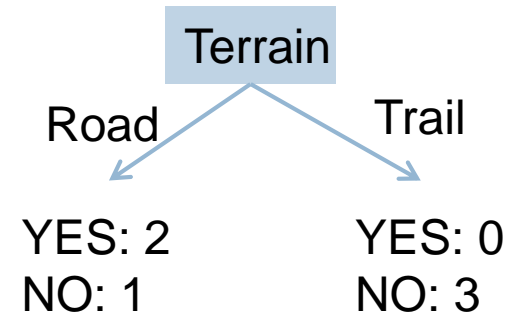
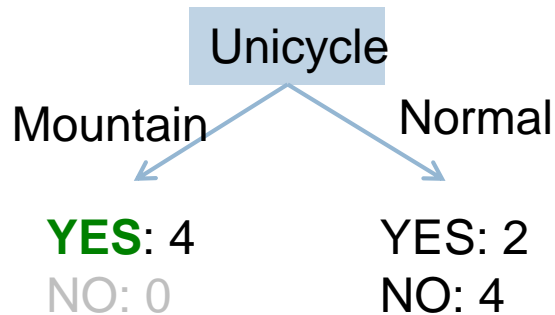
Recurse



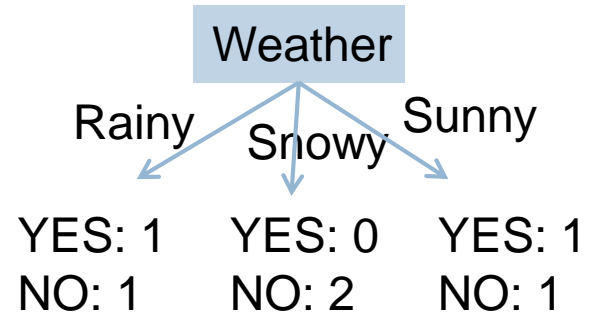
Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Mountain	Snowy	YES

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO

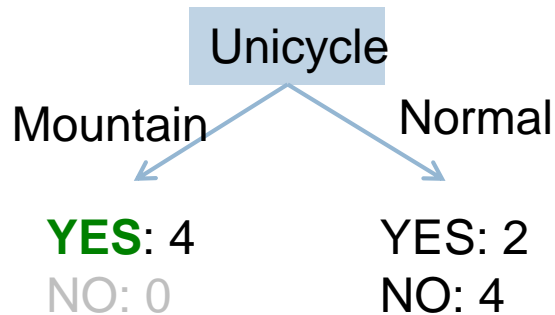
Recurse



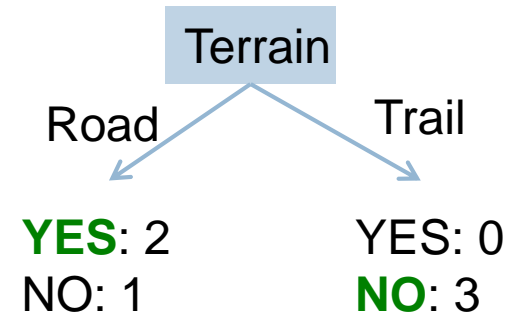
Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO



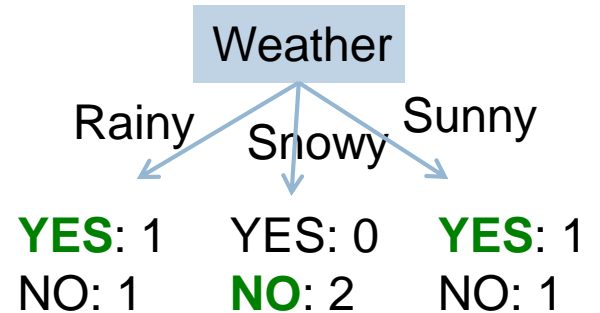
Recurse



Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO

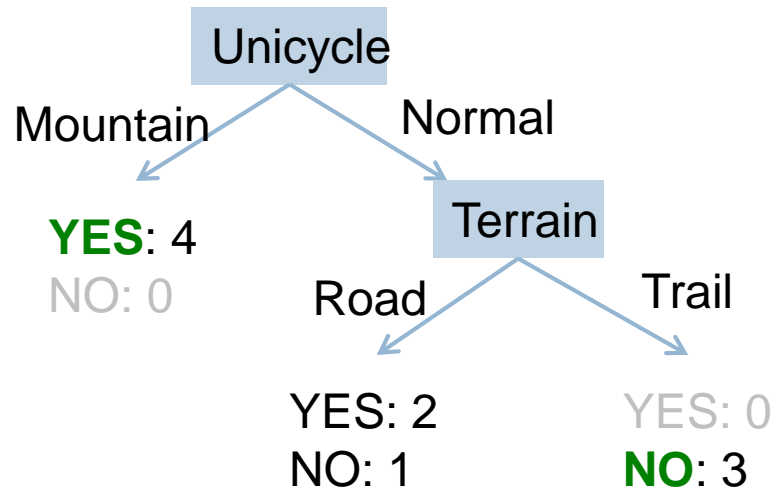


1/6



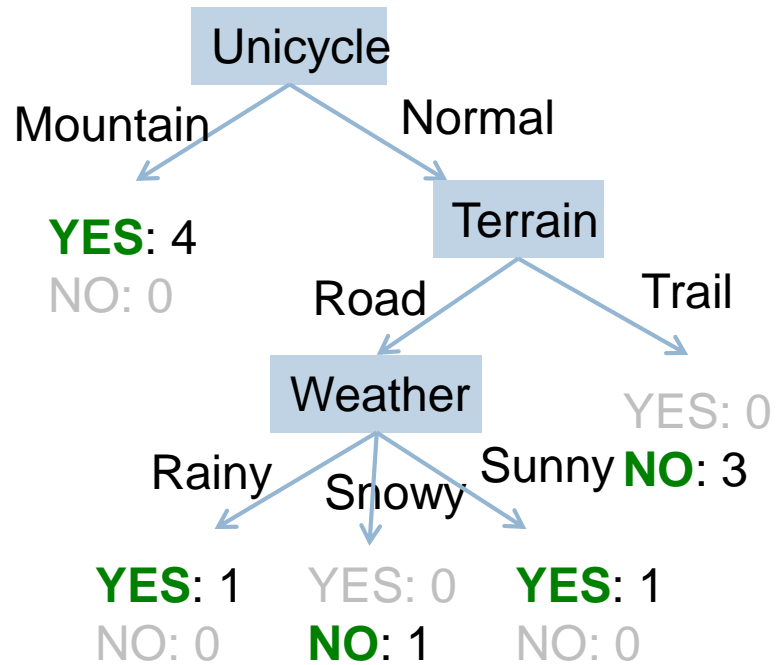
2/6

Recurse

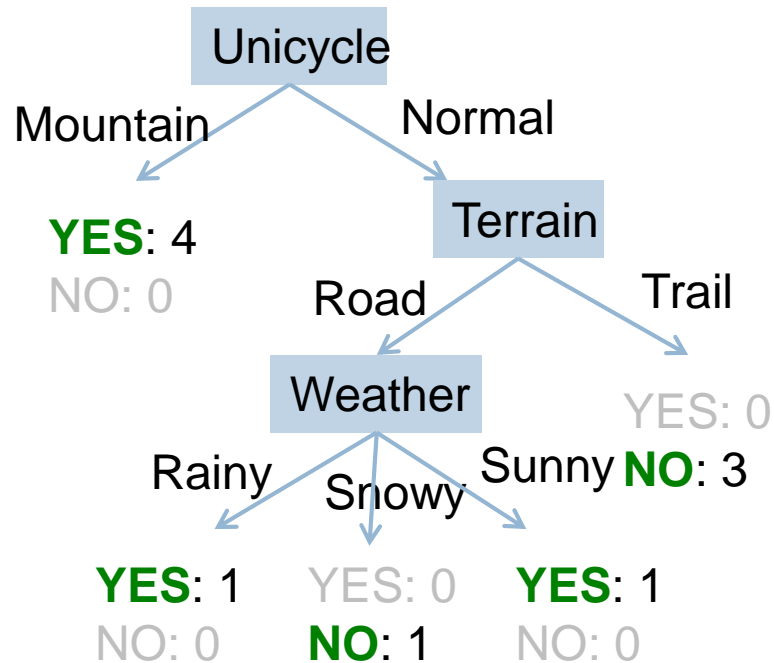


Terrain	Unicycle-type	Weather	Go-For-Ride?
Road	Normal	Sunny	YES
Road	Normal	Rainy	YES
Road	Normal	Snowy	NO

Recurse



Recurse



Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

Training error?

Are we always guaranteed to get a training error of 0?

Problematic data

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Snowy	NO
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

When can this happen?

Recursive approach

Base case: If all data belong to the same class, create a leaf node with that label **OR** all the data has the same feature values

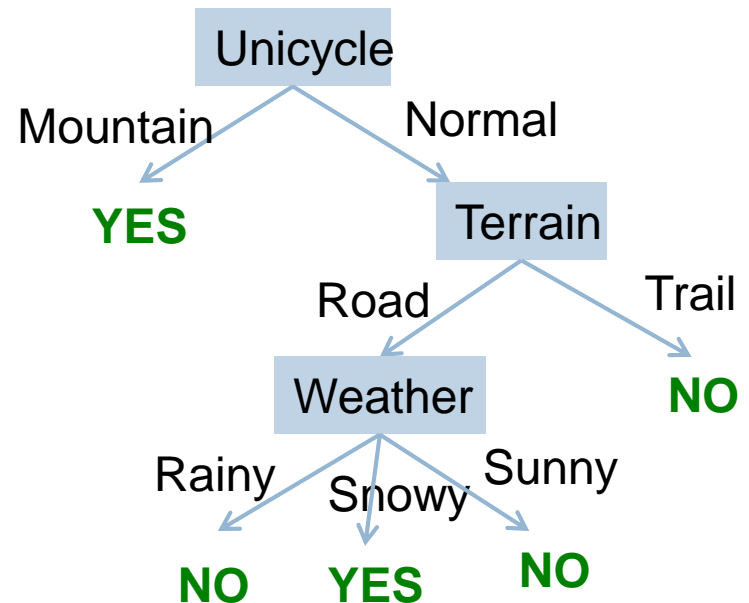
Do we always want to go all the way to the bottom?

What would the tree look like for...

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Mountain	Rainy	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Snowy	YES
Road	Mountain	Sunny	YES
Trail	Normal	Snowy	NO
Trail	Normal	Rainy	NO
Road	Normal	Snowy	YES
Road	Normal	Sunny	NO
Trail	Normal	Sunny	NO

What would the tree look like for...

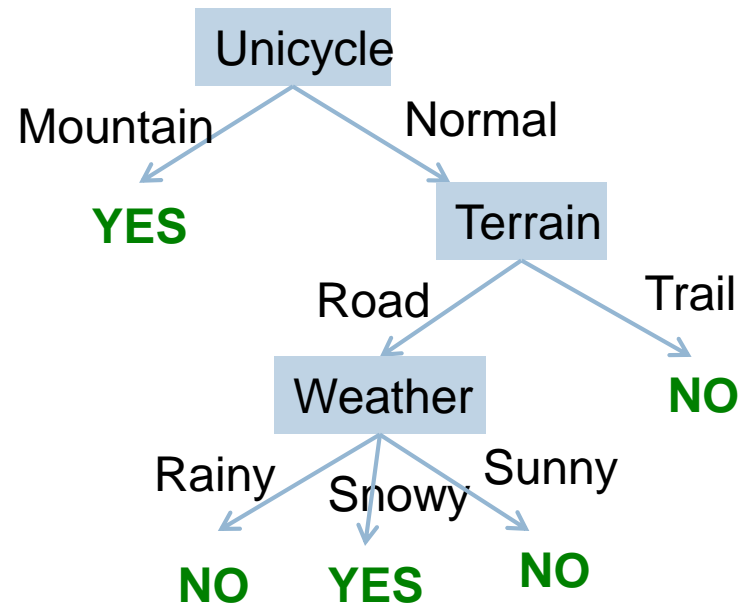
Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Mountain	Rainy	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Snowy	YES
Road	Mountain	Sunny	YES
Trail	Normal	Snowy	NO
Trail	Normal	Rainy	NO
Road	Normal	Snowy	YES
Road	Normal	Sunny	NO
Trail	Normal	Sunny	NO



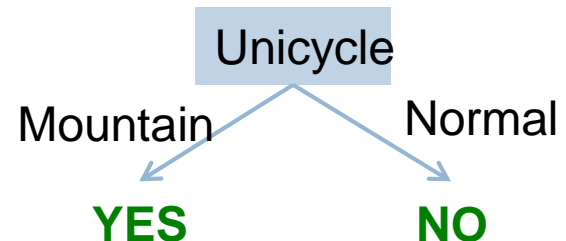
Is that what you would do?

What would the tree look like for...

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Mountain	Rainy	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Snowy	YES
Road	Mountain	Sunny	YES
Trail	Normal	Snowy	NO
Trail	Normal	Rainy	NO
Road	Normal	Snowy	YES
Road	Normal	Sunny	NO
Trail	Normal	Sunny	NO



Maybe...

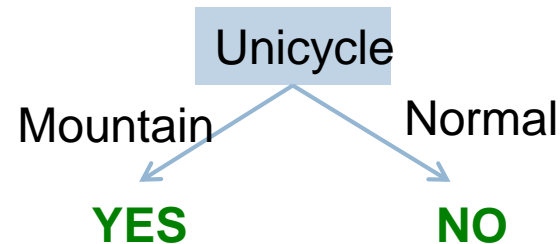


What would the tree look like for...

Terrain	Unicycle-type	Weather	Jacket	ML grade	Go-For-Ride?
Trail	Mountain	Rainy	Heavy	D	YES
Trail	Mountain	Sunny	Light	C-	YES
Road	Mountain	Snowy	Light	B	YES
Road	Mountain	Sunny	Heavy	A	YES
...	Mountain	YES
Trail	Normal	Snowy	Light	D+	NO
Trail	Normal	Rainy	Heavy	B-	NO
Road	Normal	Snowy	Heavy	C+	YES
Road	Normal	Sunny	Light	A-	NO
Trail	Normal	Sunny	Heavy	B+	NO
Trail	Normal	Snowy	Light	F	NO
...	Normal	NO
Trail	Normal	Rainy	Light	C	YES

Overfitting

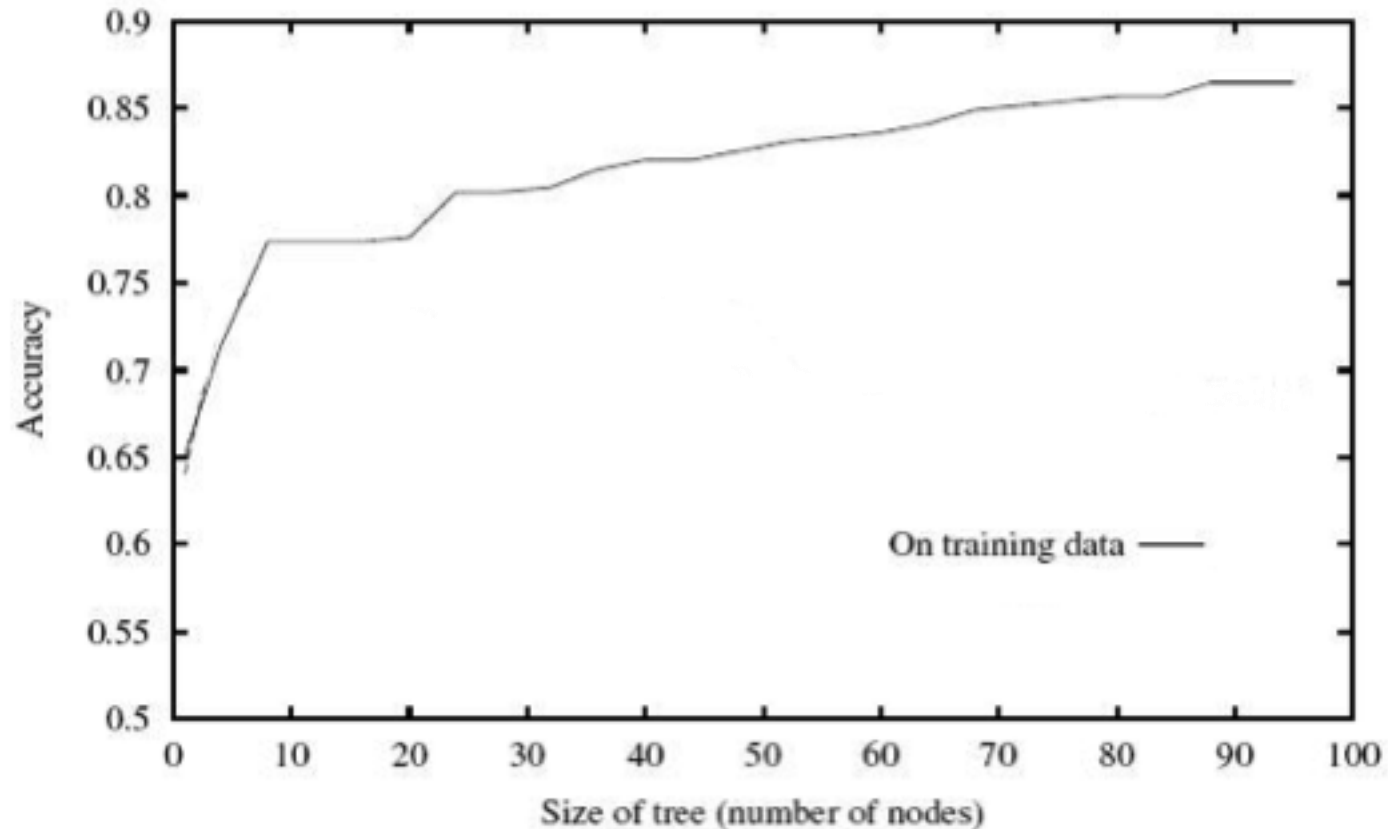
Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Mountain	Rainy	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Snowy	YES
Road	Mountain	Sunny	YES
Trail	Normal	Snowy	NO
Trail	Normal	Rainy	NO
Road	Normal	Snowy	YES
Road	Normal	Sunny	NO
Trail	Normal	Sunny	NO



Overfitting occurs when we bias our model too much towards the training data

Our goal is to learn a **general** model that will work on the training data as well as other data (i.e. test data)

Overfitting



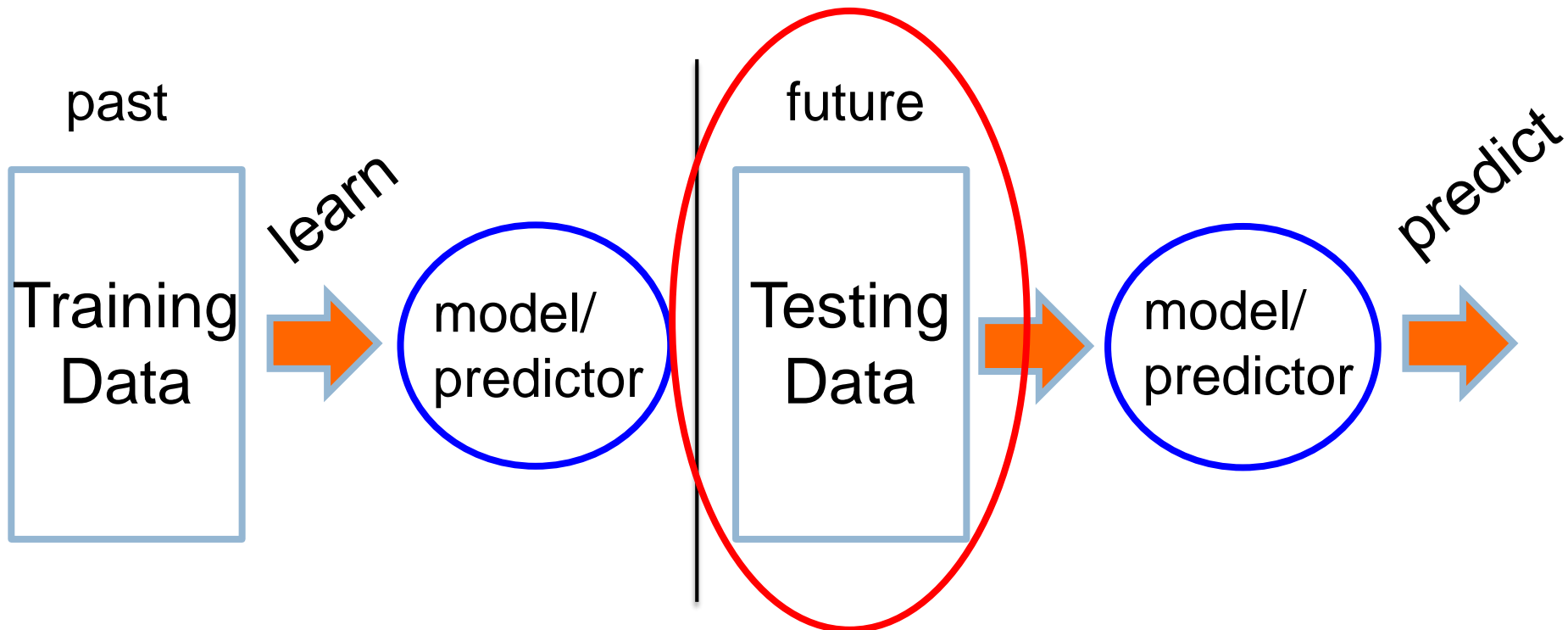
Our decision tree learning procedure always decreases training error

Is that what we want?

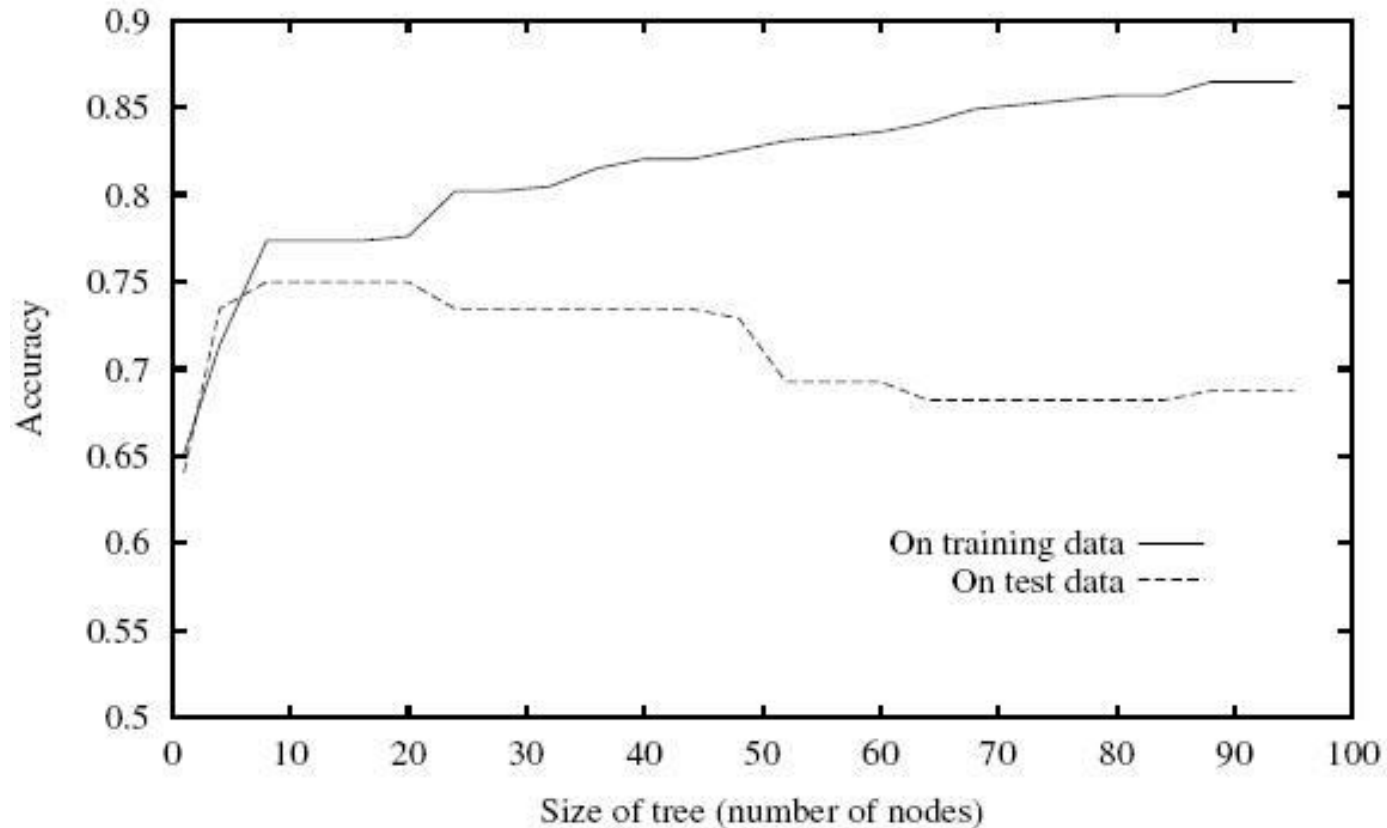
Test set error!

Machine learning is about predicting the future based on the past.

-- Hal Daume III



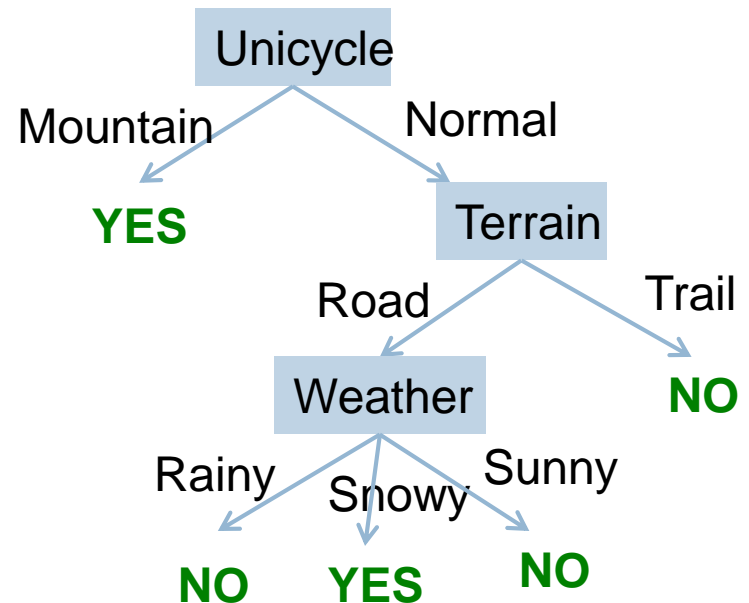
Overfitting



Even though the training error is decreasing, the testing error can go up!

Overfitting

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Mountain	Rainy	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Snowy	YES
Road	Mountain	Sunny	YES
Trail	Normal	Snowy	NO
Trail	Normal	Rainy	NO
Road	Normal	Snowy	YES
Road	Normal	Sunny	NO
Trail	Normal	Sunny	NO



How do we prevent overfitting?

Preventing overfitting

Base case: If all data belong to the same class, create a leaf node with that label **OR** all the data has the same feature values **OR**

- We've reached a particular depth in the tree
- ?

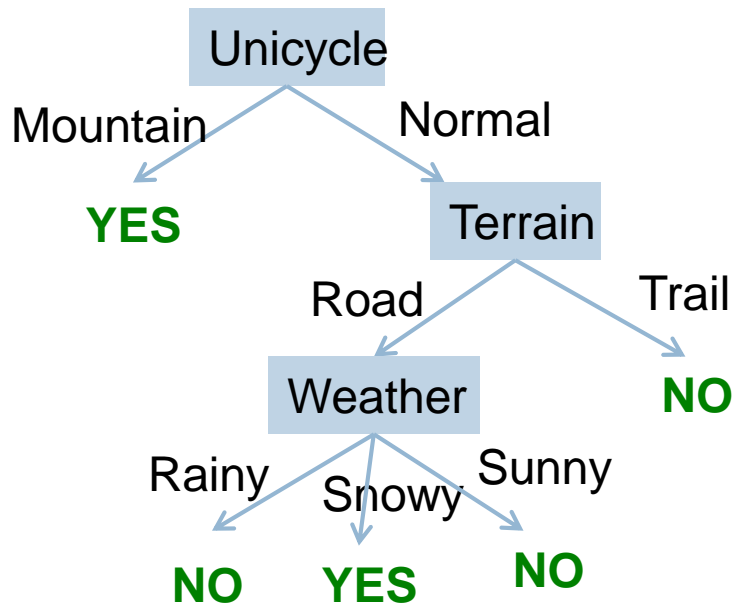
One idea: stop building the tree early

Preventing overfitting

Base case: If all data belong to the same class, create a leaf node with that label **OR** all the data has the same feature values **OR**

- We've reached a particular depth in the tree
- We only have a certain number/fraction of examples remaining
- We've reached a particular training error
- Use development data (more on this later)
- ...

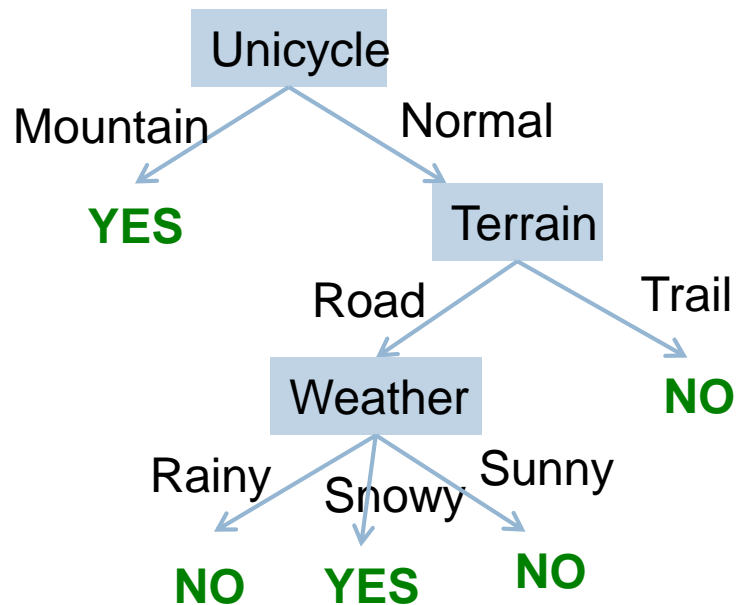
Preventing overfitting: pruning



Pruning: after the tree is built, go back and “prune” the tree, i.e. remove some lower parts of the tree

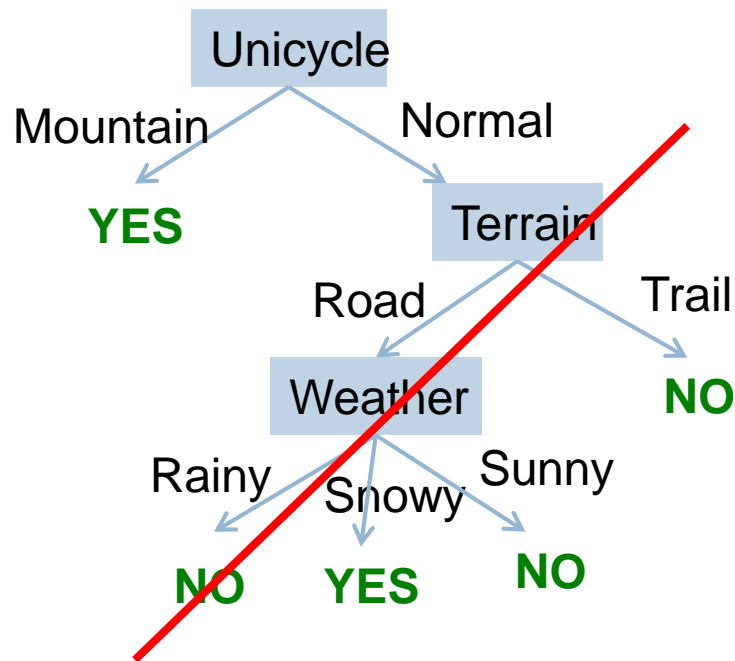
Similar to stopping early, but done after the entire tree is built

Preventing overfitting: pruning

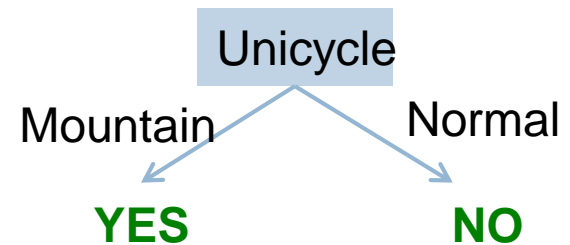


Build the full tree

Preventing overfitting: pruning

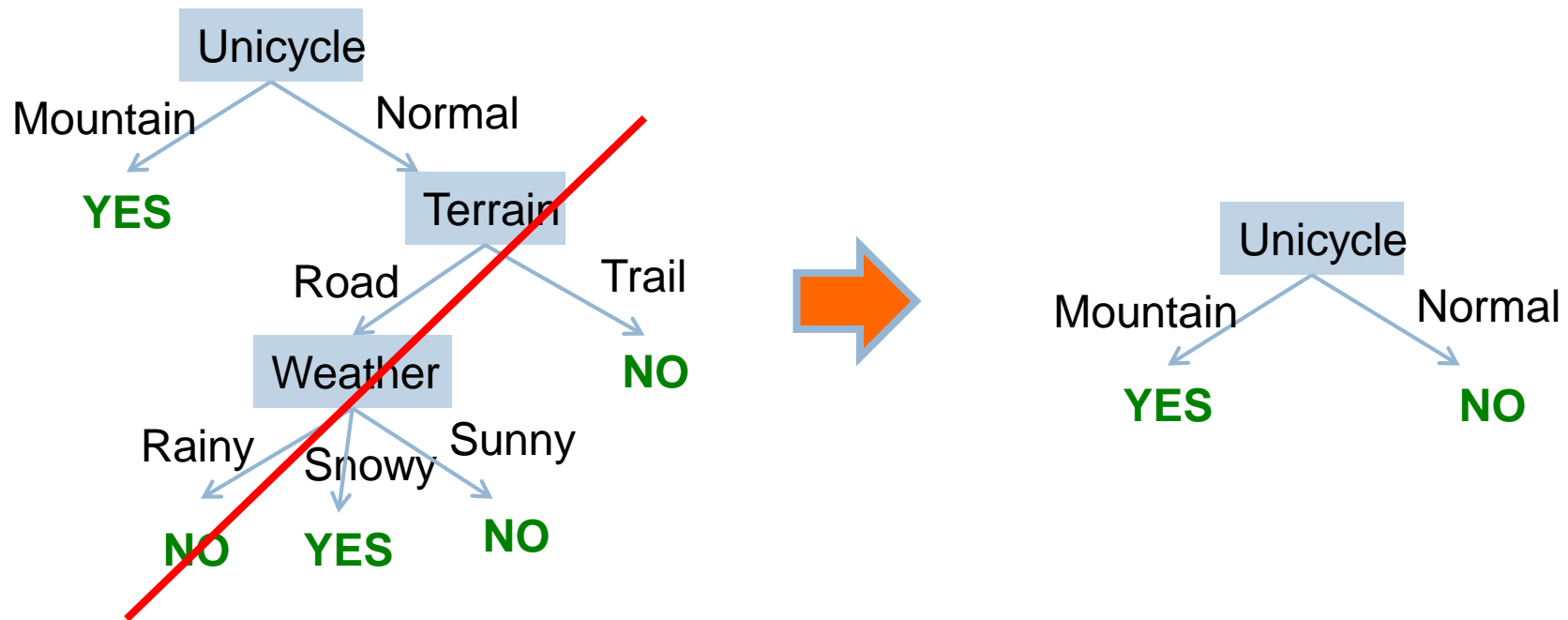


Build the full tree



Prune back leaves that are too specific

Preventing overfitting: pruning



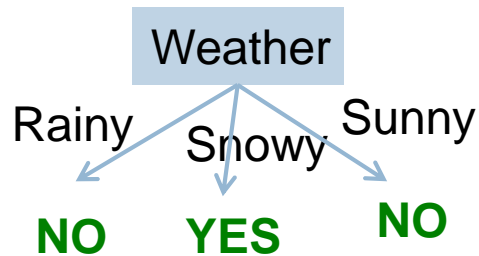
Pruning criterion?

Handling non-binary attributes

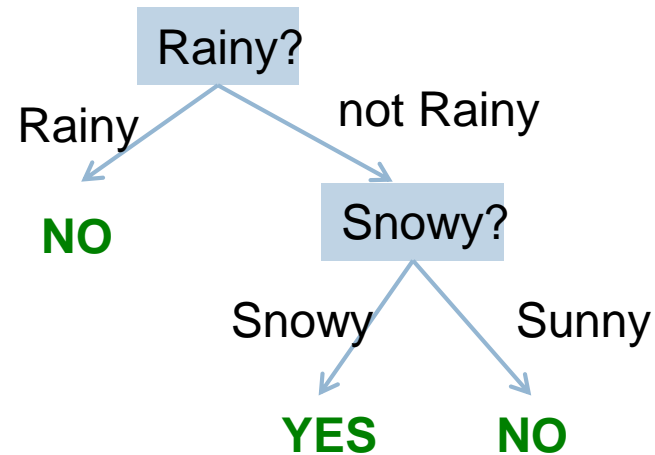
PassengerId	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	Survived	
804		3	0	0.42	0	1	2625	8.5167	0	1
756		2	0	0.67	1	1	250649	14.5	2	1
470		3	1	0.75	2	1	2666	19.2583	0	1
645		3	1	0.75	2	1	2666	19.2583	0	1
79		2	0	0.83	0	2	248738	29	2	1
832		2	0	0.83	1	1	29106	18.75	2	1
306		1	0	0.92	1	2	113781	151.55	2	1
165		3	0	1	4	1	3101295	39.6875	2	0
173		3	1	1	1	1	347742	11.1333	2	1
184		2	0	1	2	1	230136	39	2	1
382		3	1	1	0	2	2653	15.7417	0	1
387		3	0	1	5	2	2144	46.9	2	0
789		3	0	1	1	2	2315	20.575	2	1
828		2	0	1	0	2	2079	37.0042	0	1
8		3	0	2	3	1	349909	21.075	2	0
17		3	0	2	4	1	382652	29.125	1	0
120		3	1	2	4	2	347082	31.275	2	0
206		3	1	2	0	1	347054	10.4625	2	0
298		1	1	2	1	2	113781	151.55	2	0
341		2	0	2	1	1	230080	26	2	1
480		3	1	2	0	1	3101298	12.2875	2	1

What do we do with features that have multiple values?
Real-values?

Features with multiple values



Treat as an n-ary split

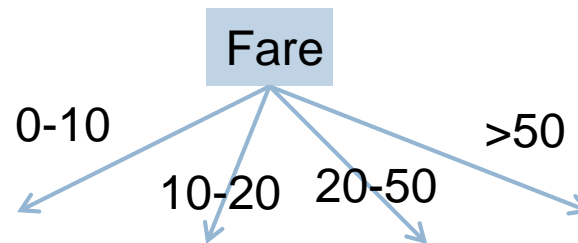
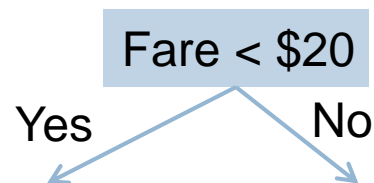


Treat as multiple binary splits

Real-valued features

Use any comparison test ($>$, $<$, \leq , \geq) to split the data into two parts

Select a range filter, i.e. $\text{min} < \text{value} < \text{max}$



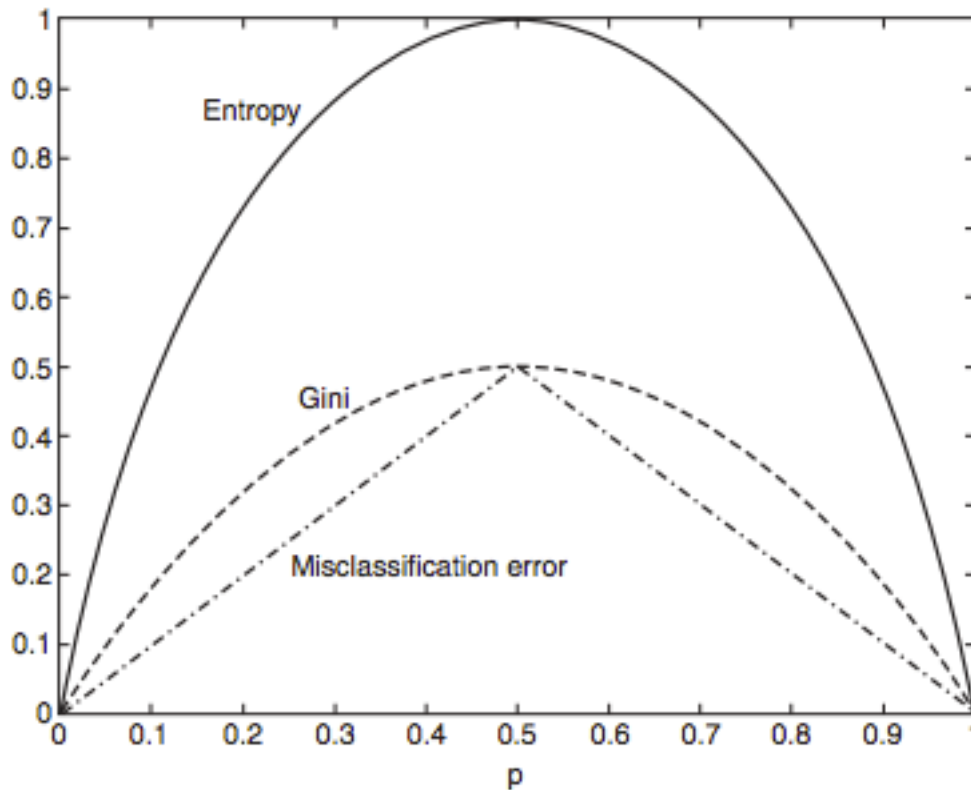
Other splitting criterion

Otherwise:

- calculate the “score” for each feature if we used it to split the data
- pick the feature with the highest score, partition the data based on that data value and call recursively

We used training error for the score. Any other ideas?

Other splitting criterion



- Entropy: how much uncertainty there is in the distribution over labels after the split
- Gini: sum of the square of the label proportions after split
- Training error = misclassification error

Decision trees

Good? Bad?



Decision trees: the good

Very intuitive and easy to interpret

Fast to run and fairly easy to implement

Historically, perform fairly well (especially with a few more tricks we'll see later on)

No prior assumptions about the data

Decision trees: the bad

Be careful with features with lots of values

Which feature
would be at
the top here?

ID	Terrain	Unicycle-type	Weather	Go-For-Ride?
1	Trail	Normal	Rainy	NO
2	Road	Normal	Sunny	YES
3	Trail	Mountain	Sunny	YES
4	Road	Mountain	Rainy	YES
5	Trail	Normal	Snowy	NO
6	Road	Normal	Rainy	YES
7	Road	Mountain	Snowy	YES
8	Trail	Normal	Sunny	NO
9	Road	Normal	Snowy	NO
10	Trail	Mountain	Snowy	YES

Decision trees: the bad

Can be problematic (slow, bad performance) with large numbers of features

Can't learn some very simple data sets (e.g. some types of linearly separable data)

Pruning/tuning can be tricky to get right

Final DT algorithm

Base cases:

1. If all data belong to the same class, pick that label
2. If all the data have the same feature values, pick majority label
3. If we're out of features to examine, pick majority label
4. If the we don't have any data left, pick majority label of *parent*
5. *If some other stopping criteria* exists to avoid overfitting, pick majority label

Otherwise:

- calculate the “score” for each feature if we used it to split the data
- pick the feature with the highest score, partition the data based on that data value and call recursively

Alternative Score Calculation – Information Gain

- Why information gain is better over accuracy?
 - ▣ Decision trees are generally prone to over-fitting and accuracy doesn't generalize well to unseen data.
 - ▣ One advantage of information gain is that -- due to the factor $-p \cdot \log(p)$ in the entropy definition -- leafs with a small number of instances are assigned less weight ($\lim_{p \rightarrow 0} -p \cdot \log(p) = 0$) and it favors dividing data into bigger but homogeneous groups.
 - ▣ This approach is usually more stable and also chooses the most impactful features close to the root of the tree.

Decision Tree - Classification

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with **decision nodes** and **leaf nodes**. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy). Leaf node (e.g., Play) represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called **root node**. Decision trees can handle both categorical and numerical data.

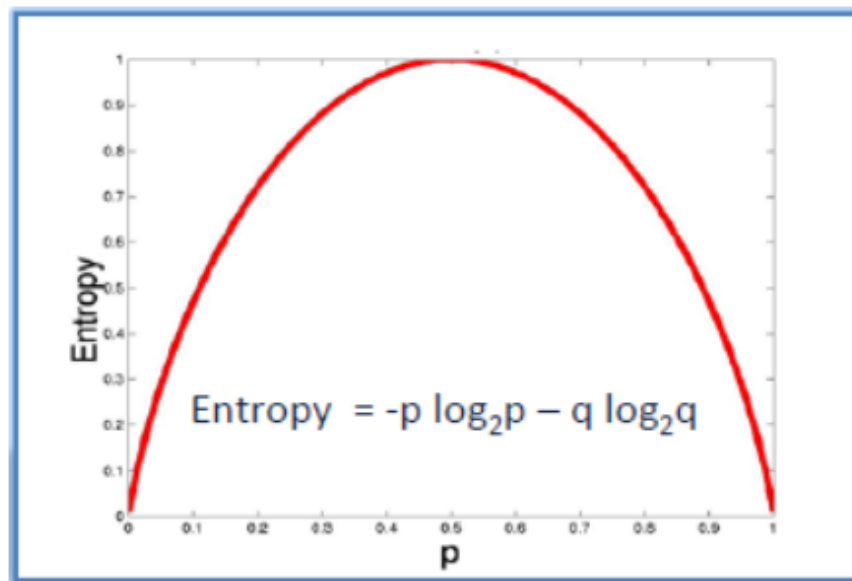


Algorithm

The core algorithm for building decision trees called **ID3** by J. R. Quinlan which employs a top-down, greedy search through the space of possible branches with no backtracking. ID3 uses *Entropy* and *Information Gain* to construct a decision tree. In ZeroR model there is no predictor, in OneR model we try to find the single best predictor, naive Bayesian includes all predictors using Bayes' rule and the independence assumptions between predictors but decision tree includes all predictors with the dependence assumptions between predictors.

Entropy

A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). ID3 algorithm uses entropy to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.



$$\text{Entropy} = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

To build a decision tree, we need to calculate two types of entropy using frequency tables as follows:

a) Entropy using the frequency table of one attribute:

$$E(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5



$$\begin{aligned}
 \text{Entropy(PlayGolf)} &= \text{Entropy}(5,9) \\
 &= \text{Entropy}(0.36, 0.64) \\
 &= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64) \\
 &= 0.94
 \end{aligned}$$

b) Entropy using the frequency table of two attributes:

$$E(T, X) = \sum_{c \in X} P(c)E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14



$$\begin{aligned}
 E(\text{PlayGolf}, \text{Outlook}) &= P(\text{Sunny}) * E(3,2) + P(\text{Overcast}) * E(4,0) + P(\text{Rainy}) * E(2,3) \\
 &= (5/14) * 0.971 + (4/14) * 0.0 + (5/14) * 0.971 \\
 &= 0.693
 \end{aligned}$$

Information Gain

The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

Step 1: Calculate entropy of the target.

$$\begin{aligned}\text{Entropy}(\text{PlayGolf}) &= \text{Entropy}(5,9) \\ &= \text{Entropy}(0.36, 0.64) \\ &= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64) \\ &= 0.94\end{aligned}$$

Step 2: The dataset is then split on the different attributes. The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the Information Gain, or decrease in entropy.

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

$$\text{Gain}(T, X) = \text{Entropy}(T) - \text{Entropy}(T, X)$$

$$\begin{aligned}\text{G}(\text{PlayGolf}, \text{Outlook}) &= \text{E}(\text{PlayGolf}) - \text{E}(\text{PlayGolf}, \text{Outlook}) \\ &= 0.940 - 0.693 = 0.247\end{aligned}$$

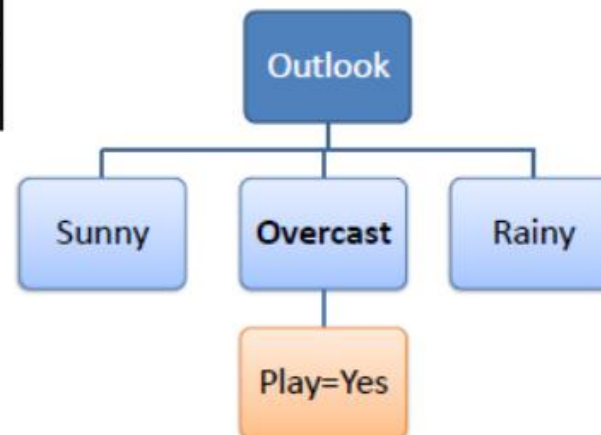
Step 3: Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch.

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
		Gain = 0.247	

Outlook	Temp	Humidity	Windy	Play Golf
Sunny	Mild	High	FALSE	Yes
	Cool	Normal	FALSE	Yes
	Cool	Normal	TRUE	No
	Mild	Normal	FALSE	Yes
	Mild	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
	Cool	Normal	TRUE	Yes
	Mild	High	TRUE	Yes
	Hot	Normal	FALSE	Yes
Rainy	Hot	High	FALSE	No
	Hot	High	TRUE	No
	Mild	High	FALSE	No
	Cool	Normal	FALSE	Yes
	Mild	Normal	TRUE	Yes

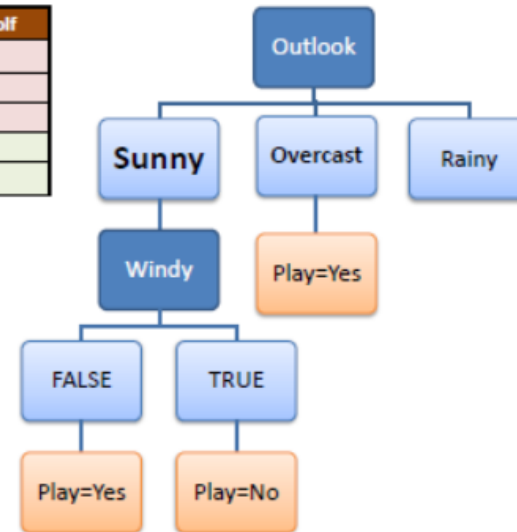
Step 4a: A branch with entropy of 0 is a leaf node.

Temp	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes



Step 4b: A branch with entropy more than 0 needs further splitting.

Temp	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No



Step 5: The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified.

Decision Tree to Decision Rules

A decision tree can easily be transformed to a set of rules by mapping from the root node to the leaf nodes one by one.

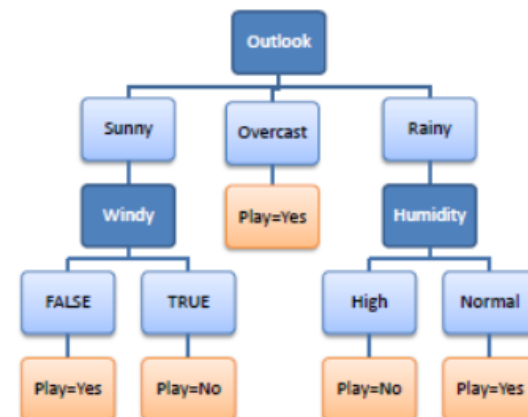
R_1 : IF (Outlook=Sunny) AND
(Windy=FALSE) THEN Play=Yes

R_2 : IF (Outlook=Sunny) AND
(Windy=TRUE) THEN Play=No

R_3 : IF (Outlook=Overcast) THEN
Play=Yes

R_4 : IF (Outlook=Rainy) AND
(Humidity=High) THEN Play=No

R_5 : IF (Outlook=Rain) AND
(Humidity=Normal) THEN
Play=Yes



- The source of above Information Gain example >

https://www.saedsayad.com/decision_tree.htm