Machine Learning

Decision Trees

Part 1: The Fundamentals

Part 2: Attribute Selection Algorithms

Part 3: Decision Tree Algorithms

Decision Trees

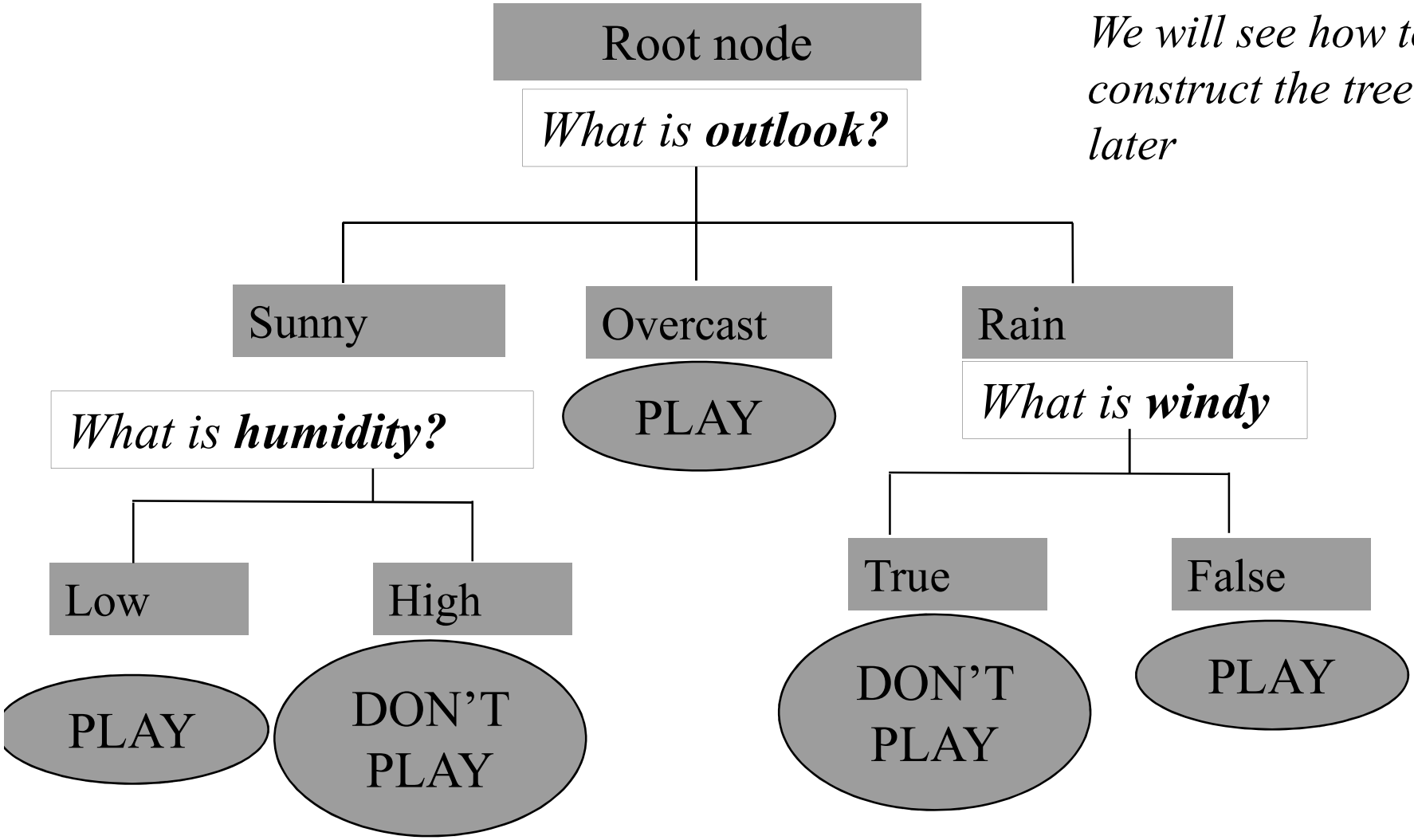
Part 1: The Fundamentals

1. What are decision trees are and how are they used to classify data?
2. What techniques could be employed to fit a decision tree from data?
3. What is a greedy algorithm?
4. What is the general structure of a greedy algorithm for building a tree?

Quinlan's classic classification example TRAINING DATA

Input Variables				Class
Outlook	Temp	Humidity	Windy	Play Golf
sunny	high (85)	high	0 (Weak)	0 (No)
sunny	high (80)	high	1 (Strong)	0
overcast	high (83)	high	0	1 (Yes)
rain	low (70)	high	0	1
rain	low (68)	high	0	1
rain	low (65)	low	1	0
overcast	low (64)	low	1	1
sunny	low (72)	high	0	0
sunny	low (69)	low	0	1
rain	low (75)	high	0	1
sunny	low (75)	low	1	1
overcast	low (72)	high	1	1
overcast	high (81)	low	0	1
rain	low (71)	high	1	0

An Example Decision Tree



We will see how to construct the tree later

Decision Tree as Rule Set

Any decision tree can be expressed as a set of mutually exclusive rules

```
if outlook =sunny
        if humidity =low        then play=true
        else                    then play=false
else if outlook =overcast       then play =true
else if outlook = rain
        if windy =true                  then play = false
        else                            then play = true
```

Decision Tree Terminology

- A decision tree is a graphical representation of non-overlapping set of rules.
- By convention the tree is drawn growing down the page.
- The **root node** is at the top.
- The **leaf** or terminal nodes are at the bottom.
- The tree partitions the input space into disjoint regions represented by the leaf nodes.

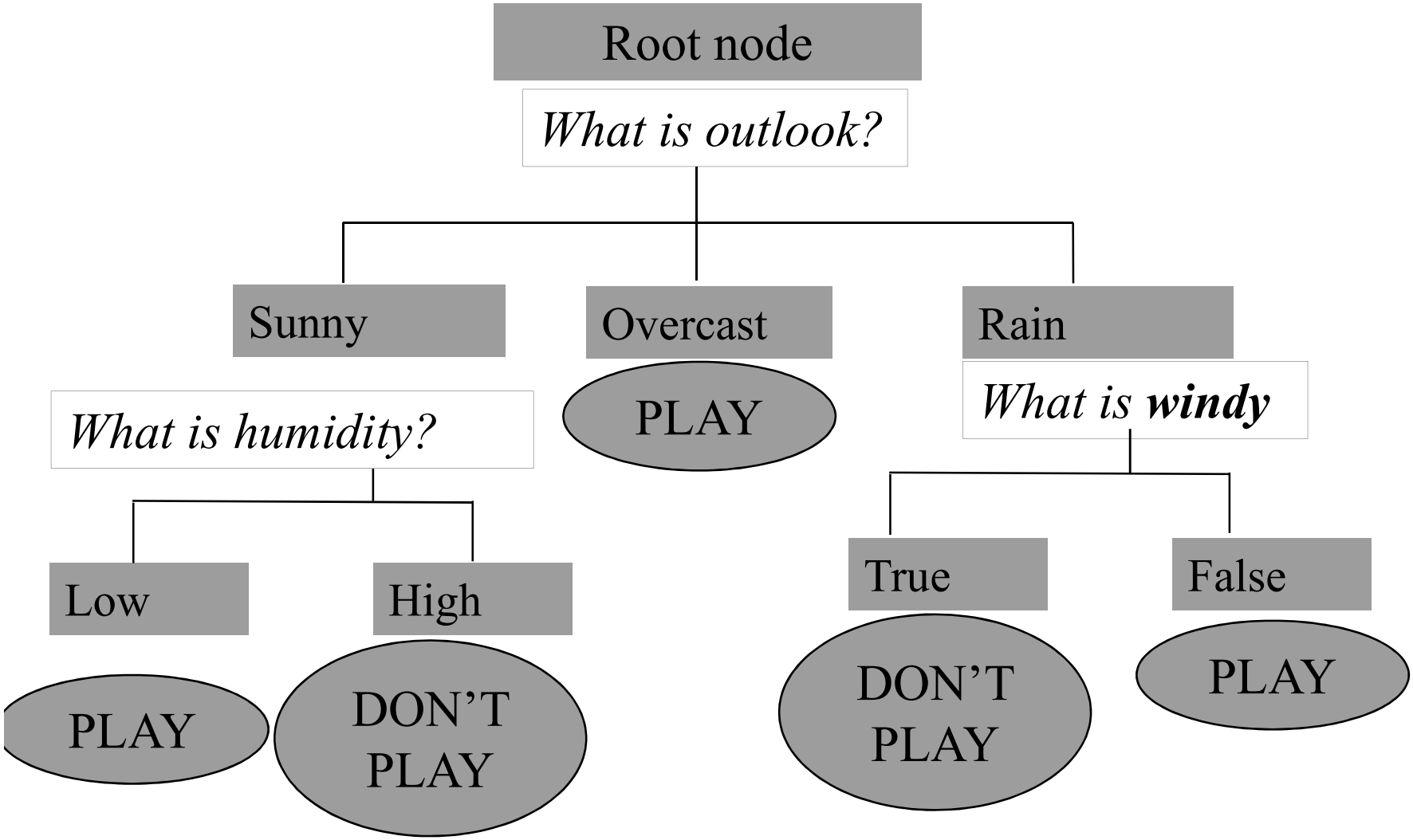
Decision Trees

- There are two potential objectives of performing classifications:
  - **Explanation**: the analysis can be used to help understand the fundamental underlying relationships in the data
  - **Prediction**: the primary concern may be to be able to accurately predict new cases
- Decision trees can be used for both tasks
- Decision tree learning is one of the most widely used techniques in machine learning
- Decision trees form the basis of the most popular ensemble algorithms

Using a Decision Tree to Classify new Cases

- Start at the root node
- Each non-terminal node contains a question.
- Pass instance on to the next node in the tree according to the answer to the question
- Continue until a terminal node is reached
- The pattern is classified according to the label of the terminal node

Its sunny, humid and windy. Do you predict that I will play golf?



Predict you will not play

Finding a Decision Tree from Training Data

- We could construct many, many different trees. Why use this particular tree?
- We like this tree because it explains the training data well.
- This means that when we run the training data through this tree, it gets the answer correct

(that does not mean it will generalise well to new data, more later!)

- We need a computationally feasible algorithm to find a tree that explains the training data.
- This is the model fitting problem

Model Fitting Strategies

Brute Force: try every possible model and select the best

Grid Search: try models evenly spaced over the parameter values

Neighbourhood Search: Start from a single solution iteratively move to nearby solutions until a criteria is met

Greedy Construction: Build the model one step at a time

Model Fitting Strategies Example

Suppose we are fitting the parameters of a linear perceptron to this model

$$w_1 x_1 + w_2 x_2$$

Brute Force — Try all possible values for the weights. Weights are continuous, so ……

Grid Search — Define a discrete number of values for each weight, then evaluate all combinations

Neighbourhood Search — Start with initial weights, then change them by small amounts to improve the model until no improvement occurs

Greedy Model — Find the best value for first weight, then add in the second variable and find the best weight

Greedy Algorithms

- A greedy algorithm is an algorithm that always takes the best immediate, or local, solution while finding an answer.
- Greedy algorithms work in phases. In each phase, a decision is made that appears to be good, without regard for future consequences.
- Generally, this means that some *local optimum* is chosen. This 'take what you can get now' strategy is the source of the name for this class of algorithms.
- We often have to employ greedy algorithms when the space of possible solutions is exponential

Travelling Salesman Problem

- Given a list of *n* cities and the distance between each of them, find the shortest route that visits every city exactly once.

An Example Greedy Algorithm (suboptimal!)

Informal Algorithm

Choose a random starting city

Always move to the closest city not yet visited

Greedy Algorithms

- Generally greedy algorithms will not enumerate the search space. Some greedy algorithms will find the optimal solution for some problems
- In many cases a greedy algorithm will not always find the best solution. It may stop in a local optimum. Because of this it is called an **Heuristic algorithm**
- Even in the cases where it may not reach the optimal we may still use a greedy algorithm as they have the benefit of being fast.
- Examples of greedy algorithms you may have come across include

  Prim's algorithm and Kruskals algorithm for Minimum Spanning Tree
  Dijkstra's algorithm for shortest path
  Huffman encoding

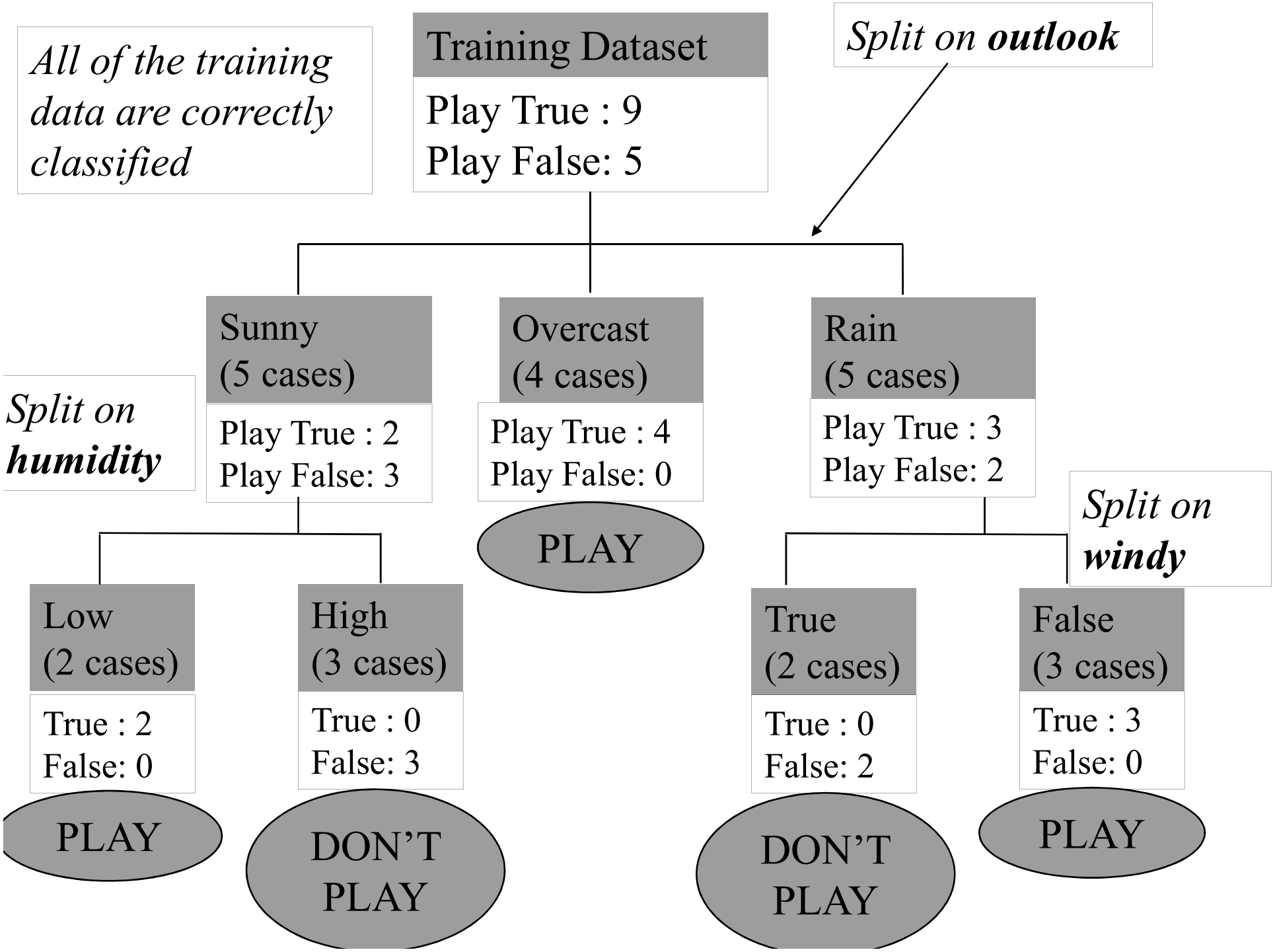
Top-Down Induction of Decision Trees

Top-Down Induction of Decision Trees (TDIDT) is a greedy heuristic recursive approach to constructing a tree

1. If all training examples at this node are classified correctly then stop.
2. Otherwise, select an attribute as the decision attribute for the split at the current node.
3. For each value of the attribute create a new child node.
4. Split training examples to child nodes.
5. Recursively apply TDIDT to patterns at each of the child nodes.

Top-Down Induction of Decision Trees

```
buildTree(DataSet D)
        TreeNode t= new TreeNode(D)
//Base case: stop building a tree
        if(stoppingCriteria(D))
            t.setAsLeaf()
            return t
//Recursive cases
//1. Choose an attribute
        Attribute A=chooseAttribute(D)
//2. Split data by attribute
        DataSet[] s= splitData(D,A)
        t.offspring= new TreeNode[size(s)]
//3. Recursively call for each split (depth first)
        for i:=1 to size(s)
                t.offspring[i]=buildTree(s[i])
        return t
```


All of the training data are correctly classified
Training Dataset
Play True : 9
Play False: 5
Split on outlook
Sunny
(5 cases)
Play True : 2
Play False: 3
Overcast
(4 cases)
Play True : 4
Play False: 0
Rain
(5 cases)
Play True : 3
Play False: 2
Split on humidity
PLAY
Split on windy
Low
(2 cases)
True : 2
False: 0
High
(3 cases)
True : 0
False: 3
True
(2 cases)
True : 0
False: 2
False
(3 cases)
True : 3
False: 0
PLAY
DON'T PLAY
DON'T PLAY
PLAY

Decision Trees

Part 2: Algorithms for Selecting Branching Attributes

1. Information gain
2. Gini-index
3. Chi-squared statistic

Three operations define a TDIDT algorithm

Stopping Criteria

Given a data set at any node, should we continue to build a tree? The simplest rule is to keep building the tree until all the data are of one class. This will probably lead to overfitting (see part 3)

Choosing an Attribute to Branch On

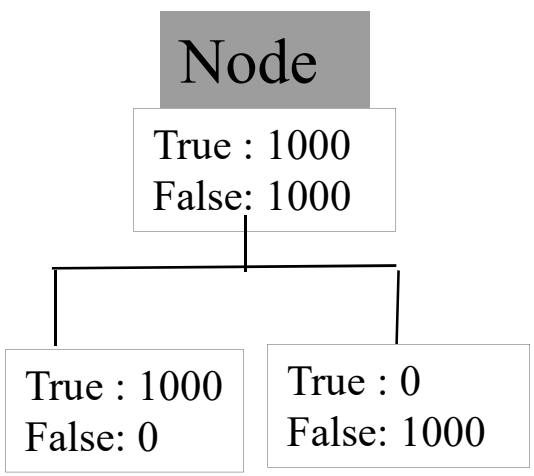
This is a key stage and we spend most of our time looking at how to do this

How to Split the Data

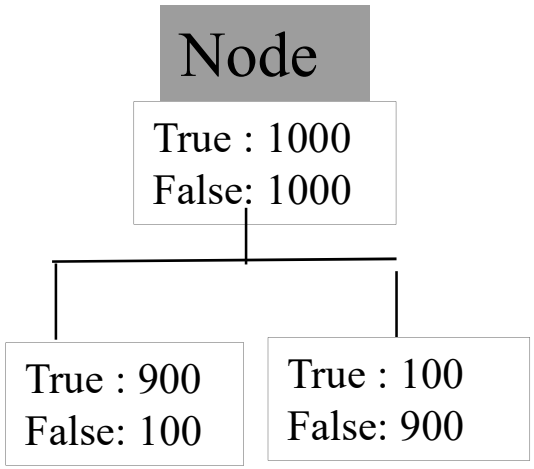
For discrete attributes, the easiest solution is to have one child node for each attribute value. However there are other ways. We also need to deal with an attribute with continuous values (e.g. temperature)

Choosing an Attribute to Branch On

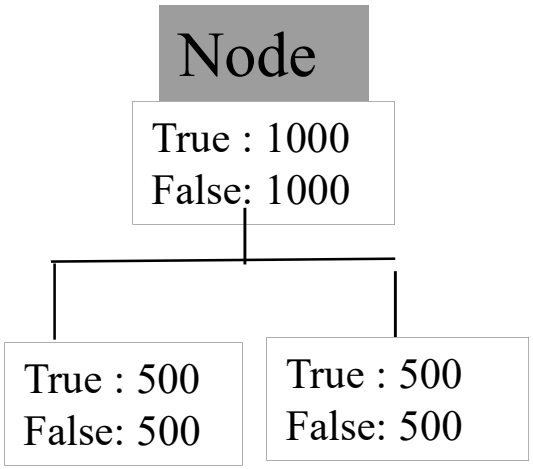
The basic principle for attribution selection is to find the attribute that leads to a branching where the classes are better split up than at the parent node



Perfect split



Good split



Useless split

Attribute Quality Measures

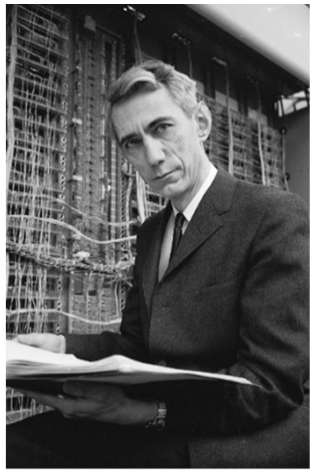
There are three commonly used algorithms for assessing how well an attribute splits class up

1. Information Gain (IG). *Based on Shannon's Entropy, it is the basis of an early decision tree, Quinlan's ID3*

2. Gini impurity. *Based on the class probability distribution. Used in the Breiman's CART family of decision trees*

3. Chi-Squared: *Based on the classic statistical test of difference of distribution, it is used in Kass's CHAID decision tree*

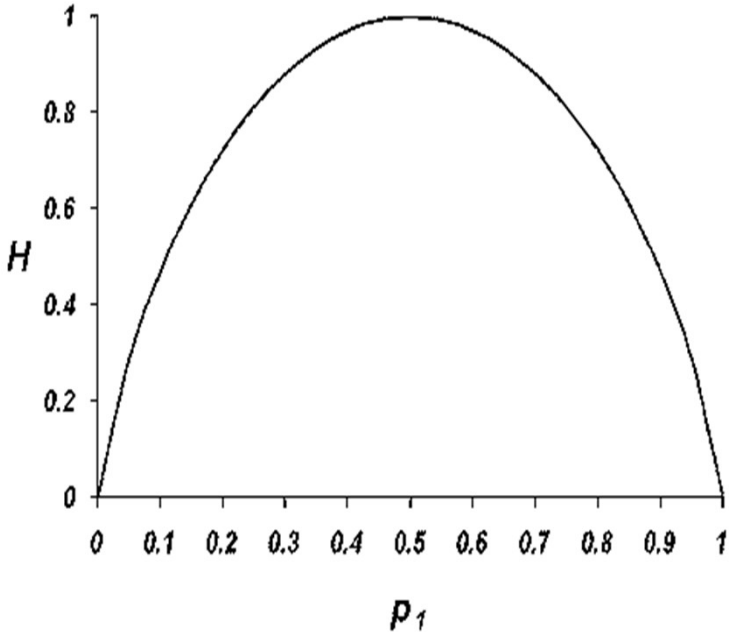
Entropy



- Claude Shannon introduced Information Theory in 1948.
- A key concept in Information Theory is Entropy.
- Entropy measures the uncertainty associated with a random variable

Imagine a coin toss.

- If the coin is double headed, there is certainty (no entropy)
- If the coin is fair, there is complete uncertainty (maximum entropy)
- If coin is biased, and is heads 75% of the time, the entropy is somewhere between the two. There is some **information** in the sequence of coin tosses



C. E. Shannon, A Mathematical Theory of Communication, Bell System Technical Journal, vol. 27, pp. 379-423, 623-656, July, October, 1948.

Entropy Function

Suppose a random variable X can take c values with probability of occurring is $p(X = i)$. The Entropy of X is defined as

$$H(X) = -\sum_{i=1}^{c} p(X = i) log_2(p(X = i))$$

H provides the lower bound of the minimum number of bits needed to encode the output of X

Probability of heads =0.5

$$H(X) = -(0.5 \log(0.5) + 0.5 \log(0.5)) = 1$$

Probability of heads =1

$$H(X) = -(1 \log(1) + 0 \log(0)) = 0$$

log(0) is undefined, but we say 0.log(0) is 0

Probability of heads =0.75

$$H(X) = -(0.75 \log(0.75) + 0.25 \log(0.25)) = 0.81$$

Entropy For a Node

Entropy is a measure of how "pure" a node is.
We estimate the probabilities as the proportion of each target value
If the class is evenly split, it is the worst scenario (high entropy)

Node X	
True : 1000 False: 1000	P(True) : 1/2 P(False): 1/2

$$H(X) = -(0.5\log(0.5) + 0.5\log(0.5)) = 1$$

If all the elements in a node are of one class, it is the best scenario (no entropy)

Node Y	
True : 2000 False: 0	P(True) : 1 P(False): 0

$$H(Y) = -(1\log(1) + 0\log(0)) = 0$$

Other splits will range between 0 and 1 (for a 2 class problem)

Node Z	
True : 9 False: 4	P(True) : 9/13 P(False): 4/13

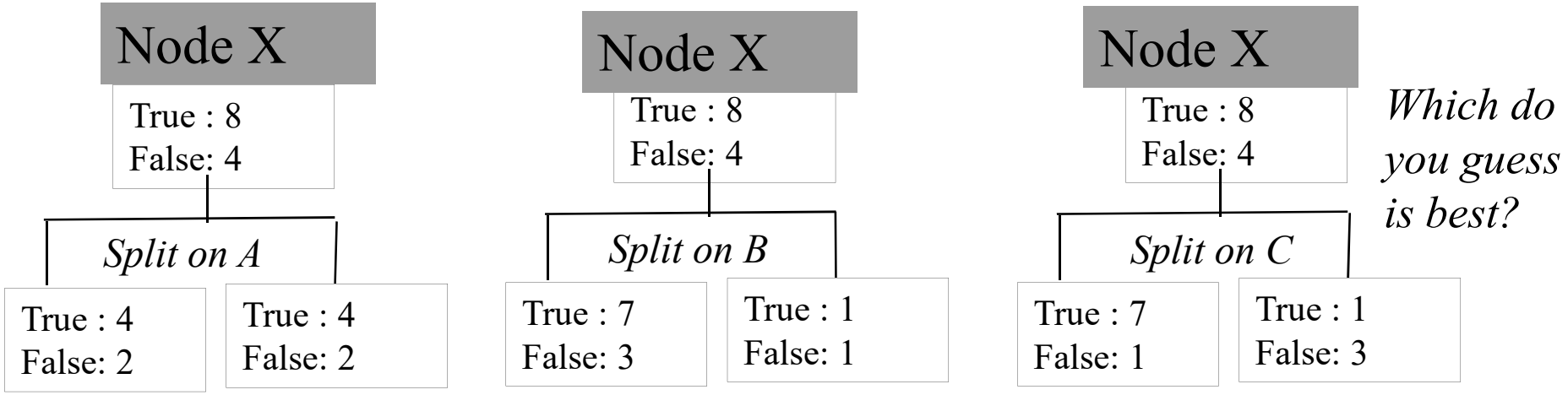
$$H(Z) = \left(\frac{9}{13}\log\left(\frac{9}{13}\right) + \frac{4}{13}\log\left(\frac{4}{13}\right)\right) = 0.89$$

Information Gain

Suppose we need to choose between three attributes that produce the following splits of the train data

The entropy at the root is

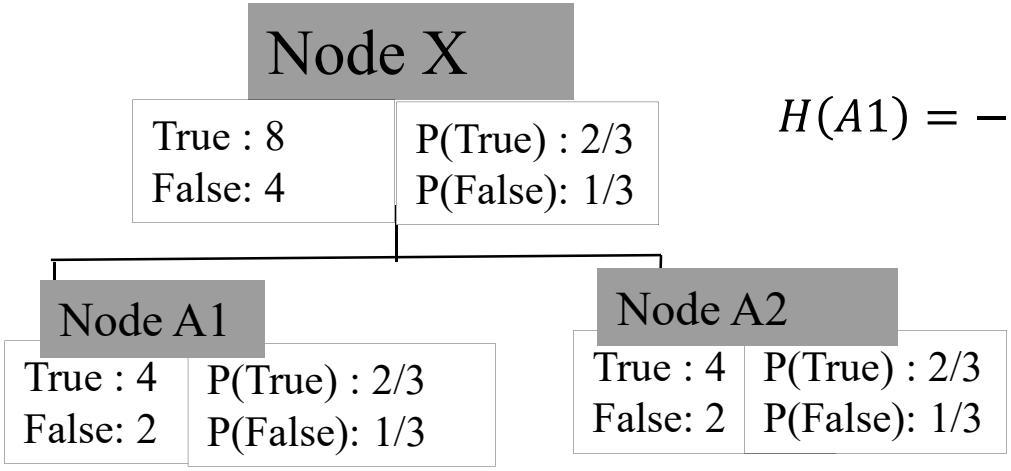
$$H(X) = -\left(\frac{8}{12}\log\left(\frac{8}{12}\right) + \frac{4}{12}\log\left(\frac{4}{12}\right)\right) = 0.9183$$



We wish to choose the split that reduces the entropy the most, or, conversely, gives the best **information gain**

Information Gain Attribute A

$$H(X) = -\left(\frac{2}{3}\log\left(\frac{2}{3}\right) + \frac{1}{3}\log\left(\frac{1}{3}\right)\right) = 0.9183$$



$$H(A1) = -\left(\frac{2}{3}\log\left(\frac{2}{3}\right) + \frac{1}{3}\log\left(\frac{1}{3}\right)\right) = 0.9183$$

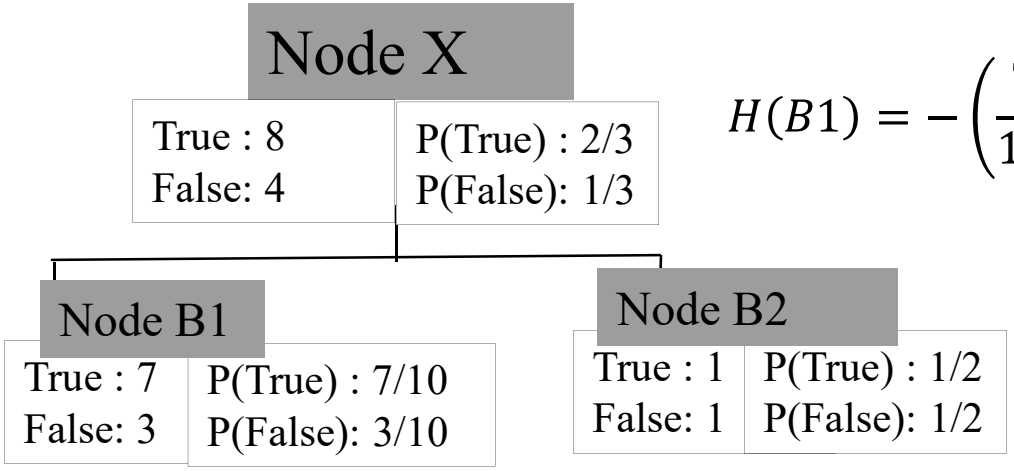
H(A2)= 0.9183

The distribution at each child is identical, so the split adds nothing

$$gain(X, B) = H(X) - \frac{6}{12}H(A1) - \frac{6}{12}H(A2) = 0$$

Information Gain Attribute B

$$H(X) = -\left(\frac{2}{3}\log\left(\frac{2}{3}\right) + \frac{1}{3}\log\left(\frac{1}{3}\right)\right) = 0.9183$$



$$H(B1) = -\left(\frac{7}{10}\log\left(\frac{7}{10}\right) + \frac{3}{10}\log\left(\frac{3}{10}\right)\right) = 0.8813$$

H(B2)= 1

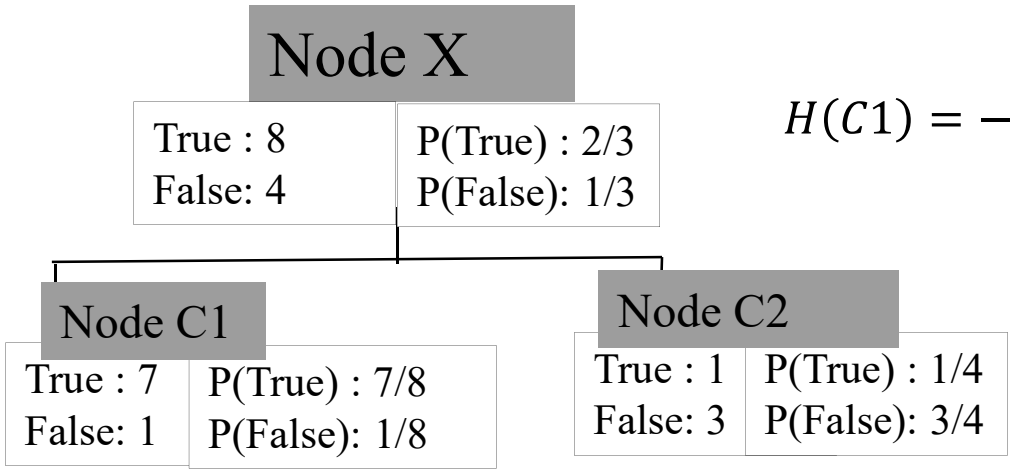
One node gives a reduction in entropy, the other doesn't.
Overall, is it a good split?

There are more cases in node B1, **so it gets greater weight.**
Weight is proportional to number of cases at each child node

$$gain(X, B) = H(X) - \frac{10}{12}H(B1) - \frac{2}{12}H(B2) = 0.01722$$

Information Gain Attribute C

$$H(X) = -\left(\frac{2}{3}\log\left(\frac{2}{3}\right) + \frac{2}{3}\log\left(\frac{2}{3}\right)\right) = 0.9183$$



$$H(C1) = -\left(\frac{7}{8}\log\left(\frac{7}{8}\right) + \frac{1}{8}\log\left(\frac{1}{8}\right)\right) = 0.5436$$

H(C2)= 0.8113

Both children have lower entropy than the parent. C1 is a good indicator of "True", C2 of "False".

$$gain(X, C) = H(X) - \frac{8}{12}H(B1) - \frac{4}{12}H(B2) = 0.2855$$

Attribute C gives the best information gain, so we choose this one to branch on

Information Gain Definition

$$\text{Gain}(X, A) = H(X) - \sum_{Y \in V} \frac{|Y|}{|X|} H(Y)$$

- *H(X)* is the entropy of the parent node
- A is the attribute we are assessing
- $|X|$ is the number of cases at node *X*
- $|Y|$ is the number of cases at node *Y*

$$V = \{A_1, A_2, \ldots, A_k\}$$

- *V* is the set of nodes for each of the *k* possible values for attribute *A*

Play Golf Example with Information Gain

Root Node X

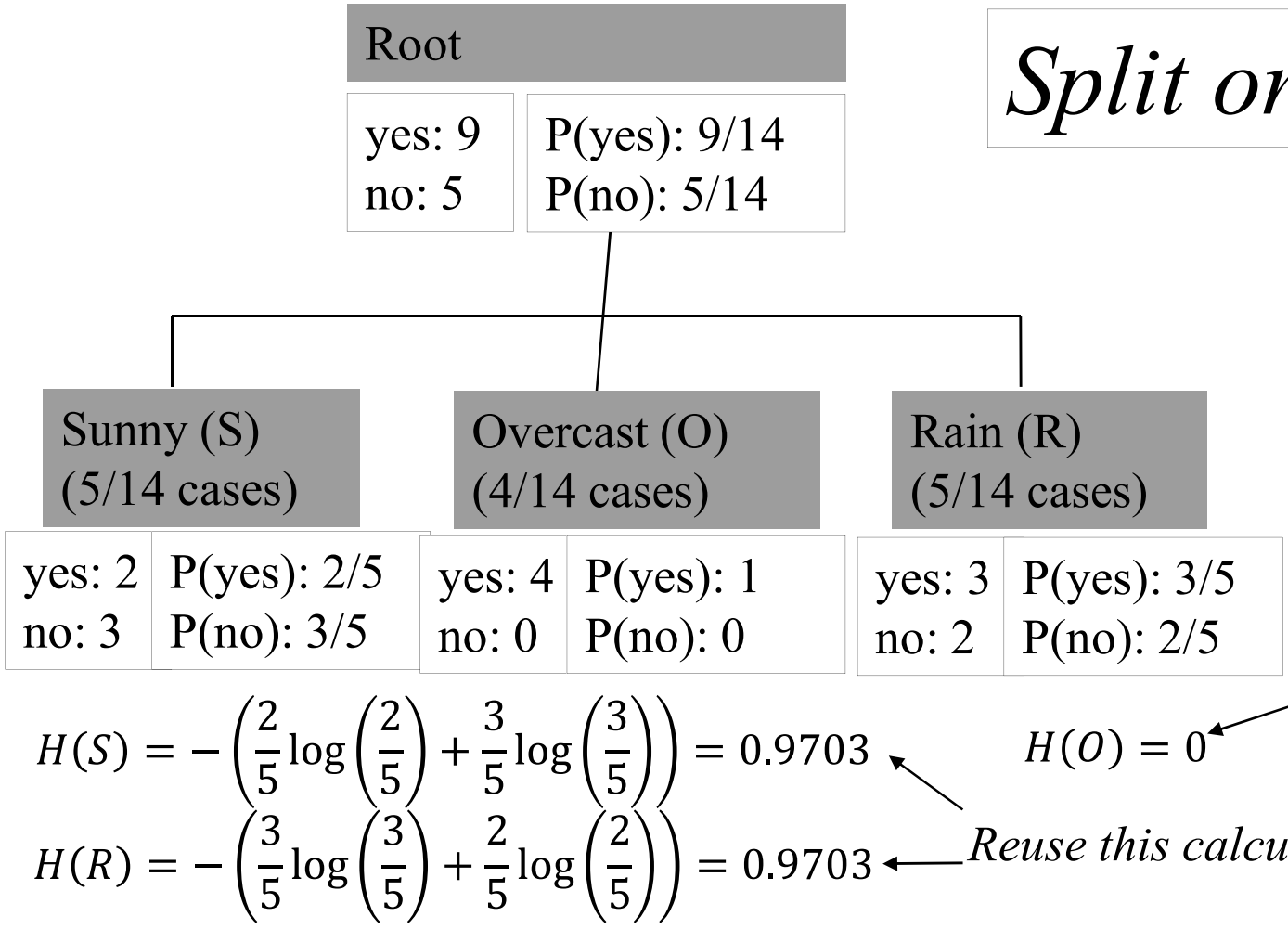
Yes: 9 No : 5	P(True) : 9/14 P(False): 5/14

$$H(X) = -\left(\frac{9}{14}\log\left(\frac{9}{14}\right) + \frac{5}{14}\log\left(\frac{5}{14}\right)\right) = 0.94$$

We need to assess which of the four attributes gives us the best gain

Input Variables				Class
Outlook	Temp	Humidity	Windy	Play Golf
sunny	high (85)	high	0 (Weak)	0 (No)
sunny	high (80)	high	1 (Strong)	0
overcast	high (83)	high	0	1 (Yes)
rain	low (70)	high	0	1
rain	low (68)	high	0	1
rain	low (65)	low	1	0
overcast	low (64)	low	1	1
sunny	low (72)	high	0	0
sunny	low (69)	low	0	1
rain	low (75)	high	0	1
sunny	low (75)	low	1	1
overcast	low (72)	high	1	1
overcast	high (81)	low	0	1
rain	low (71)	high	1	0

*Split on **outlook**?*



Certainty always has entropy 0.

$$H(O) = 0$$

$$H(S) = -\left(\frac{2}{5}\log\left(\frac{2}{5}\right) + \frac{3}{5}\log\left(\frac{3}{5}\right)\right) = 0.9703$$

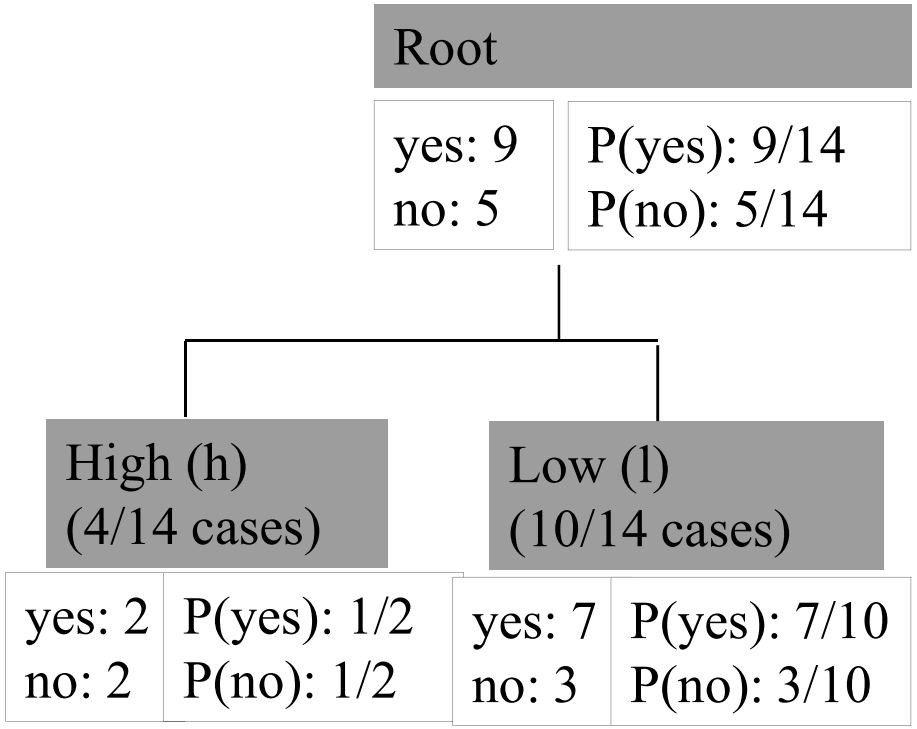
$$H(R) = -\left(\frac{3}{5}\log\left(\frac{3}{5}\right) + \frac{2}{5}\log\left(\frac{2}{5}\right)\right) = 0.9703$$

Reuse this calculation

$$\text{Gain}(X, Outlook) = H(X) - \sum_{Y \in V} \frac{|Y|}{|X|} H(Y)$$

=0.94-(5/14 H(S)+4/14H(O)+5/14H(R))

=0.24675



*Split on **temp**?*

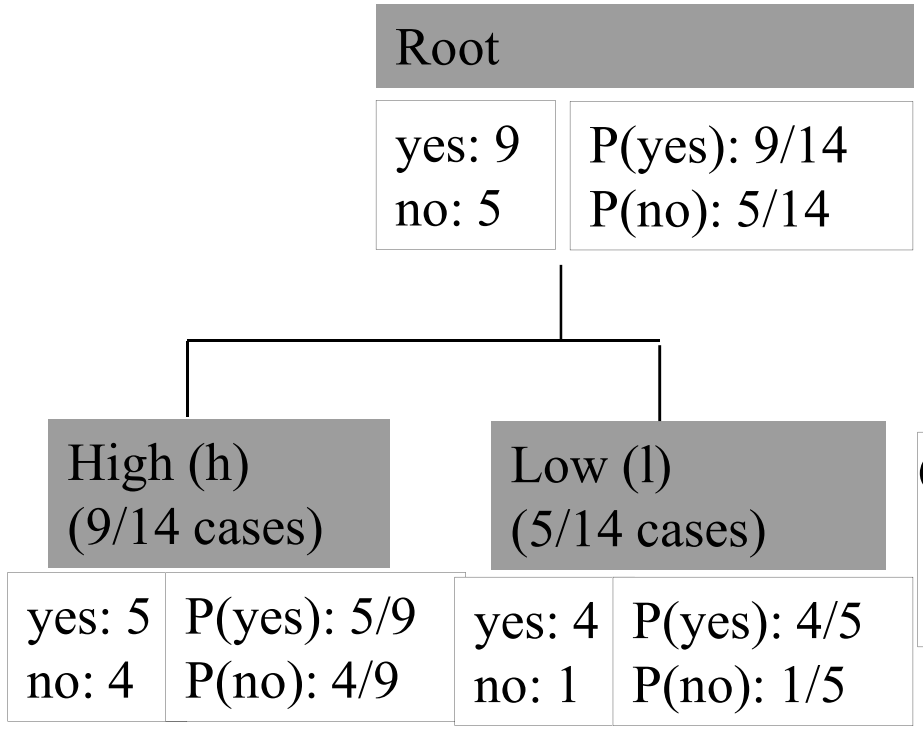
Even split always has entropy 1 (for 2 class problems)

$$H(h) = 1$$

$$H(l) = -\left(\frac{7}{10}\log\left(\frac{7}{10}\right) + \frac{3}{10}\log\left(\frac{3}{10}\right)\right) = 0.8813$$

$$\text{Gain}(X, temp) = 0.94 - \left(\frac{4}{14}H(h) + \frac{10}{14}H(l)\right)$$

$$= 0.025$$

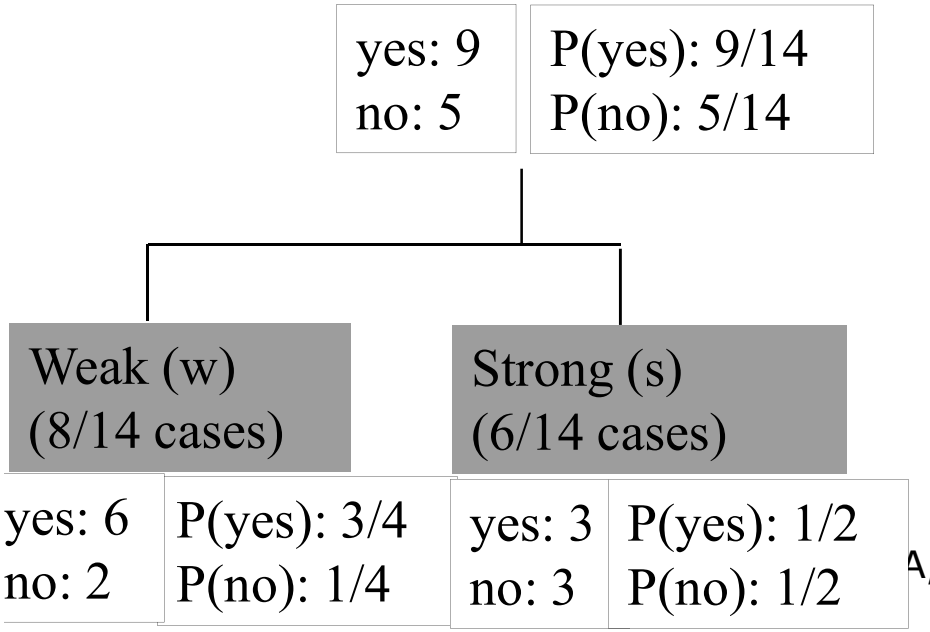


*Split on **humidity?***

$$H(h) = 0.991$$

$$H(l) = 0.7219$$

$$\text{Gain}(X, humidity) = 0.94 - \left(\frac{9}{14}H(h) + \frac{5}{14}H(l)\right) = 0.0453$$



*Split on **windy?***

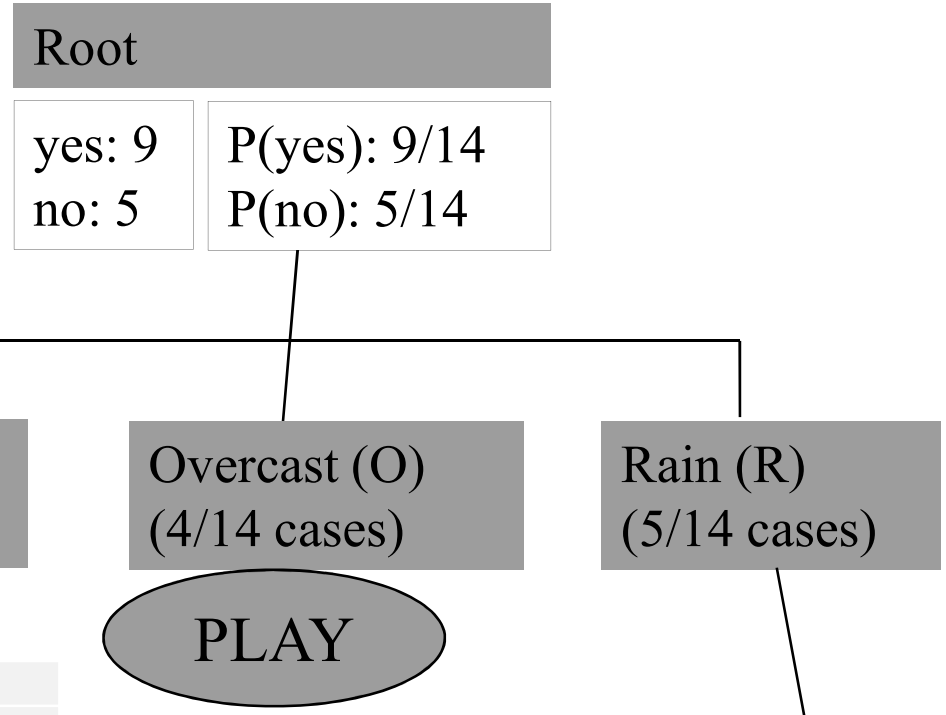
$$H(w) = 0.8113$$

$$H(s) = 1$$

$$\text{Gain}(X, windy) = 0.94 - \left(\frac{8}{14}H(h) + \frac{6}{14}H(l)\right) = 0.0481$$

$$\text{Gain}(X, outlook) = 0.24675$$
$$\text{Gain}(X, temp) = 0.025$$
$$\text{Gain}(X, humidity) = 0.0453$$
$$\text{Gain}(X, windy) = 0.0481$$

Choose Outlook



Input Variables				Class
Outlook	Temp	Humidity	Windy	Play Golf
sunny	high (85)	high	0 (Weak)	0 (No)
sunny	high (80)	high	1 (Strong)	0
sunny	low (72)	high	0	0
sunny	low (69)	low	0	1
sunny	low (75)	low	1	1

Input Variables				Class
Outlook	Temp	Humidity	Windy	Play Golf
rain	low (70)	high	0	1
rain	low (68)	high	0	1
rain	low (65)	low	1	0
rain	low (75)	high	0	1
rain	low (71)	high	1	0

*Split on **temp***

Sunny	
yes: 2 no: 3	P(yes): 2/5 P(no): 3/5

$H(sunny) = 0.9703$

High (h) (2/5 cases)	
yes: 0 no: 2	P(yes): 0 P(no): 1

Low (l) (3/5 cases)	
yes: 2 no: 1	P(yes): 2/3 P(no): 1/3

$H(h) = 0$ $H(l) = 0.971$

$\text{Gain}(sunny, temp) = 0.9703 - \left(\frac{2}{5}H(h) + \frac{3}{5}H(l)\right) = 0.42$

*Split on **humidity***

$\text{Gain}(sunny, humidity) = 0.9703$

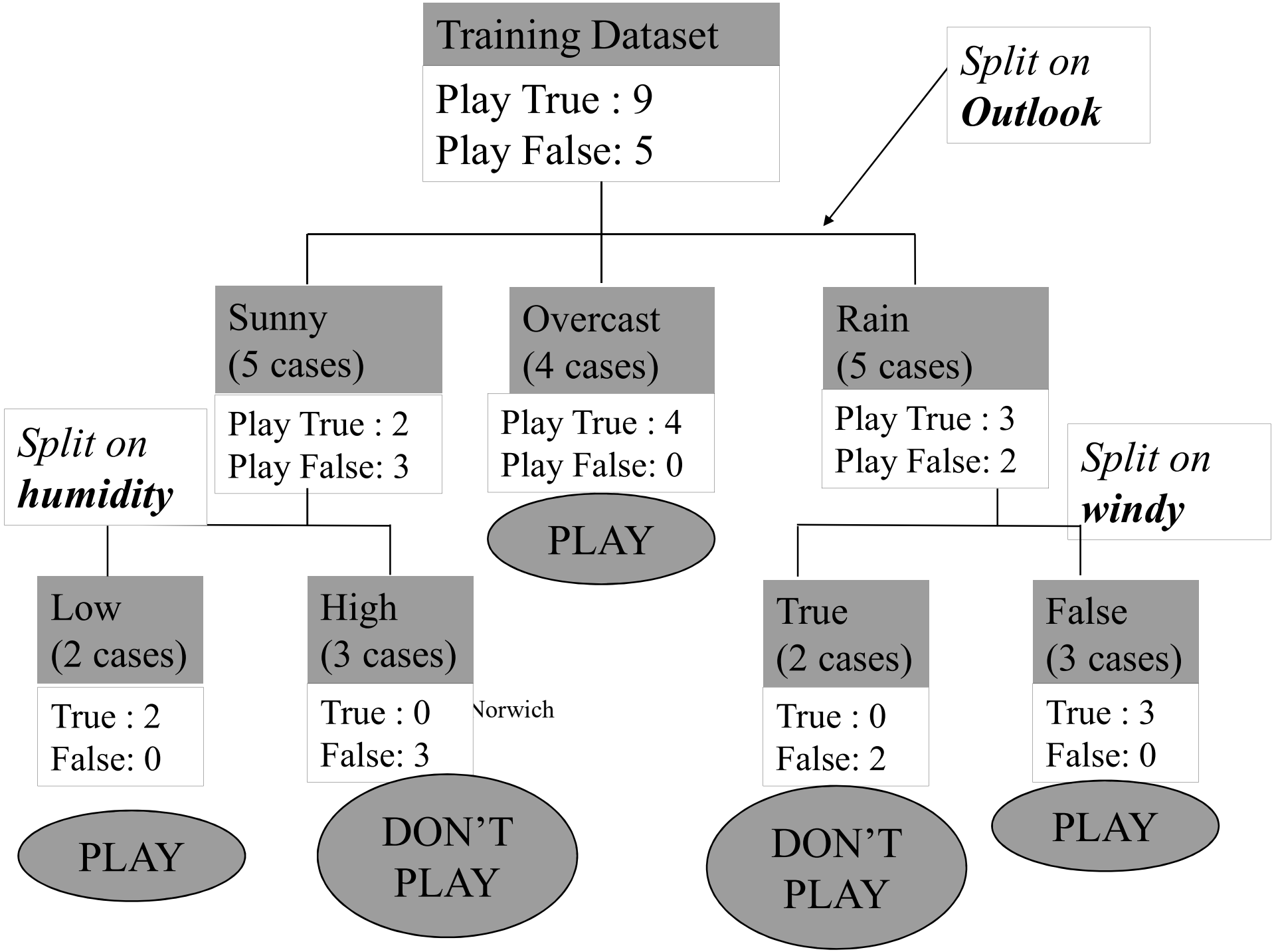
High (h) (3/5 cases)	
yes: 0 no: 3	P(yes): 0 P(no): 1

Low (l) (2/5 cases)	
yes: 2 no: 0	P(yes): 1 P(no): 0

$\text{Gain}(sunny, windy) = 0.9703 - (3/5H(w) + 2/5H(s)) = 0.02$

Input Variables				Class
Outlook	Temp	Humidity	Windy	Play Golf
sunny	high (85)	high	0 (Weak)	0 (No)
sunny	high (80)	high	1 (Strong)	0
sunny	low (72)	high	0	0
sunny	low (69)	low	0	1
sunny	low (75)	low	1	1

$\text{Gain}(sunny, temp) = 0.42$
$\text{Gain}(sunny, humidity) = 0.9703$
$\text{Gain}(sunny, windy) = 0.02$


Training Dataset
Play True : 9
Play False: 5
Split on
Outlook
Sunny
(5 cases)
Play True : 2
Play False: 3
Overcast
(4 cases)
Play True : 4
Play False: 0
Rain
(5 cases)
Play True : 3
Play False: 2
Split on
humidity
Split on
windy
Low
(2 cases)
True : 2
False: 0
High
(3 cases)
True : 0
False: 3
True
(2 cases)
True : 0
False: 2
False
(3 cases)
True : 3
False: 0
PLAY
PLAY
DON'T PLAY
DON'T PLAY
PLAY

2. Gini Impurity Measure

The Gini measure is a simple way of summarising a probability distribution
If a node has a “pure” distribution, it is good

Node

True : 1000
False: 1000

P(True) =1/2
P(False)=1/2

This is “impure” because the probabilities are the same

Node

True : 0
False: 1000

P(True) =0
P(False)=1

This is “pure” because the probabilities are all 1 or 0

Node

True : 400
False: 600

P(True) =4/10
P(False)=6/10

This has low purity because the probabilities are about the same

Node

True : 100
False: 900

P(True) =1/10
P(False)=9/10

This has high purity because it has large and small probabilities

Gini Impurity Measure

Given a random variable X with c possible values and probabilities $p(X = 1), p(X = 2), \ldots, p(X = c)$

The Gini impurity measure is defined as

$$\mathrm{I}(X) = 1 - \sum_{i=1}^{c} p(X = i)^2$$

A	P(True) =1/2 P(False)=1/2	$\mathrm{I}(A) = 1 - 0.5^2 - 0.5^2 = 0.5$	impure
B	P(True) =0 P(False)=1	$\mathrm{I}(X) = 1 - 0^2 - 1^2 = 0$	pure
C	P(True) =4/10 P(False)=6/10	$\mathrm{I}(X) = 1 - 0.4^2 - 0.6^2 = 0.48$	low purity
D	P(True) =1/10 P(False)=9/10	$\mathrm{I}(X) = 1 - 0.1^2 - 0.9^2 = 0.18$	high purity

Gini Criteria Definition

Choose the attribute with the highest Gini Score

$$\text{Gini}(X, A) = I(X) - \sum_{Y \in V} \frac{|Y|}{|X|} I(Y)$$

- *I(X)* is the impurity of the parent node
- A is the attribute we are assessing
- $|X|$ is the number of cases at node *X*
- $|Y|$ is the number of cases at node *Y*

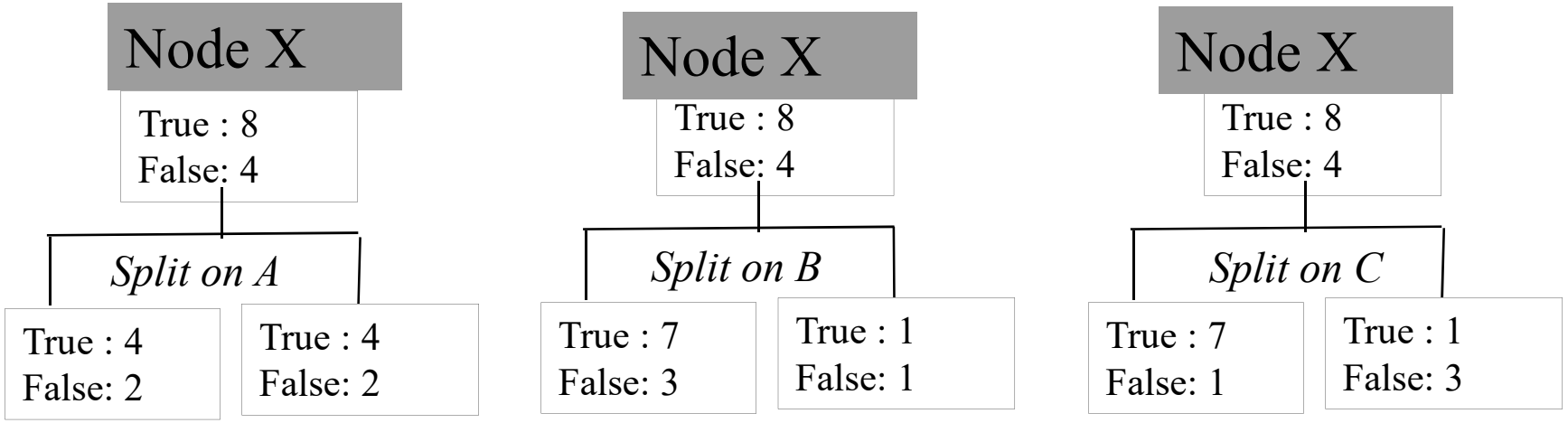
$$V = \{A_1, A_2, \ldots, A_k\}$$

- *V* is the set of nodes for each of the *k* possible values for attribute *A*

Gini Example

Suppose we need to choose between three attributes that produce the following splits of the train data

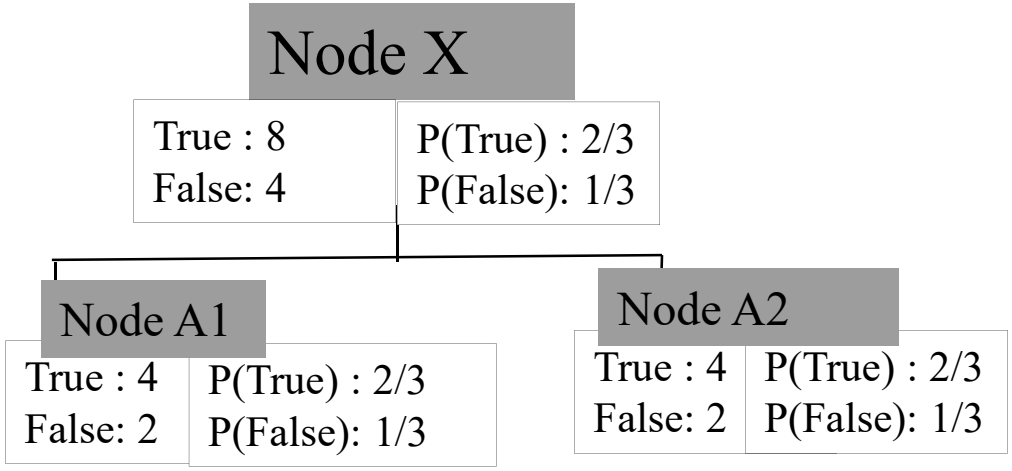
The Impurity at the root is $$\mathrm{I}(X) = 1 - \left(\frac{8}{12}^2 + \frac{4}{12}^2\right) = 0.444$$



We wish to choose the split that reduces the impurity the most, or, conversely, gives the highest **gini measure**

Gini Attribute A

$$\mathrm{I}(X) = 1 - \left(\frac{8}{12}^2 + \frac{4}{12}^2\right) = 0.444$$



$$\mathrm{I}(A1) = 0.444$$

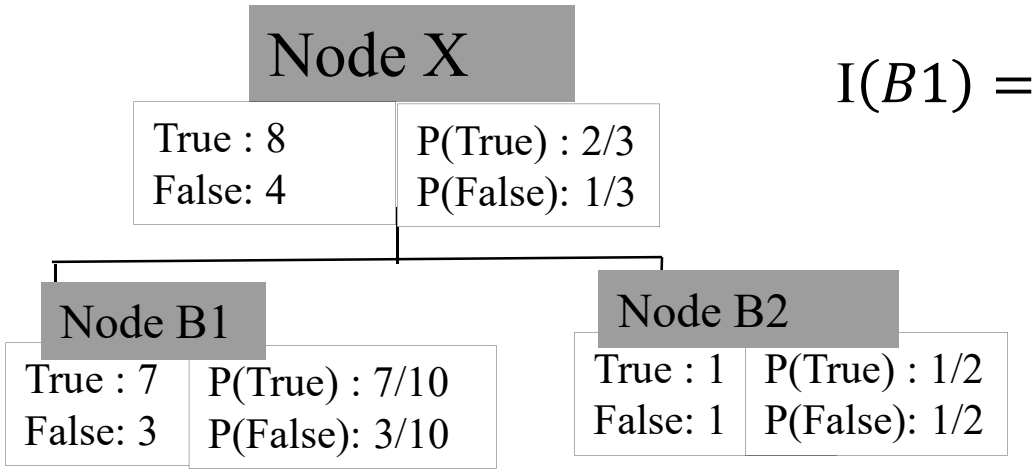
$$\mathrm{I}(A2) = 0.444$$

The split does not change the class distribution for either node, so is pointless

$$gini(X, A) = I(X) - \frac{6}{12} I(A1) - \frac{6}{12} I(A2) = 0$$

Gini Attribute B

$$\mathrm{I}(X) = 1 - \left(\frac{8}{12}^2 + \frac{4}{12}^2\right) = 0.444$$



$$\mathrm{I}(B1) = 1 - \left(\left(\frac{7}{10}\right)^2 + \left(\frac{3}{10}\right)^2\right) = 0.42$$

$$\mathrm{H}(B2) = 0.5$$

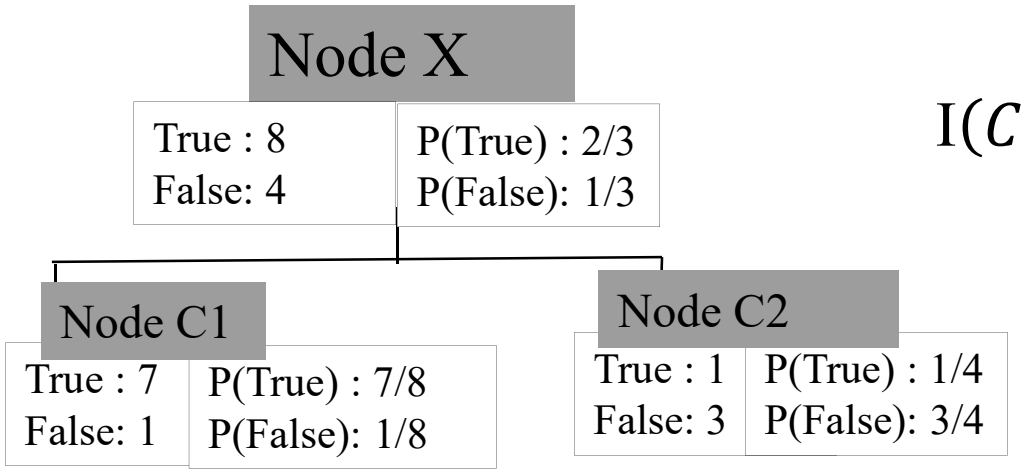
One node gives a reduction in impurity, the other doesn't.
Overall, is it a good split?

There are more cases in node B1, **so it gets greater weight.**
Weight is proportional to number of cases at each child node

$$gini(X,B) = I(X) - \frac{10}{12}I(B1) - \frac{2}{12}I(B2) = 0.0111$$

Information Gain Attribute C

$$\mathrm{I}(X) = 1 - \left(\frac{8}{12}^2 + \frac{4}{12}^2\right) = 0.444$$



$$\mathrm{I}(C1) = 1 - \left(\frac{7^2}{8} + \frac{1^2}{8}\right) = 0.22$$

I(C2)= 0.375

Both children have lower impurity than the parent. C1 is a good indicator of "True", C2 of "False".

$$gini(X, C) = I(X) - \frac{7}{8}I(B1) - \frac{1}{8}I(B2) = 0.1736$$

Attribute C gives the best gini, so we choose this one to branch on

Information Gain or Gini?

The measures are very similar

$$H(X) = -\sum_{i=1}^{c} p(X = i) log_2(p(X = i))$$

$$\text{Gain}(X, A) = H(X) - \sum_{Y \in V} \frac{|Y|}{|X|} H(Y)$$

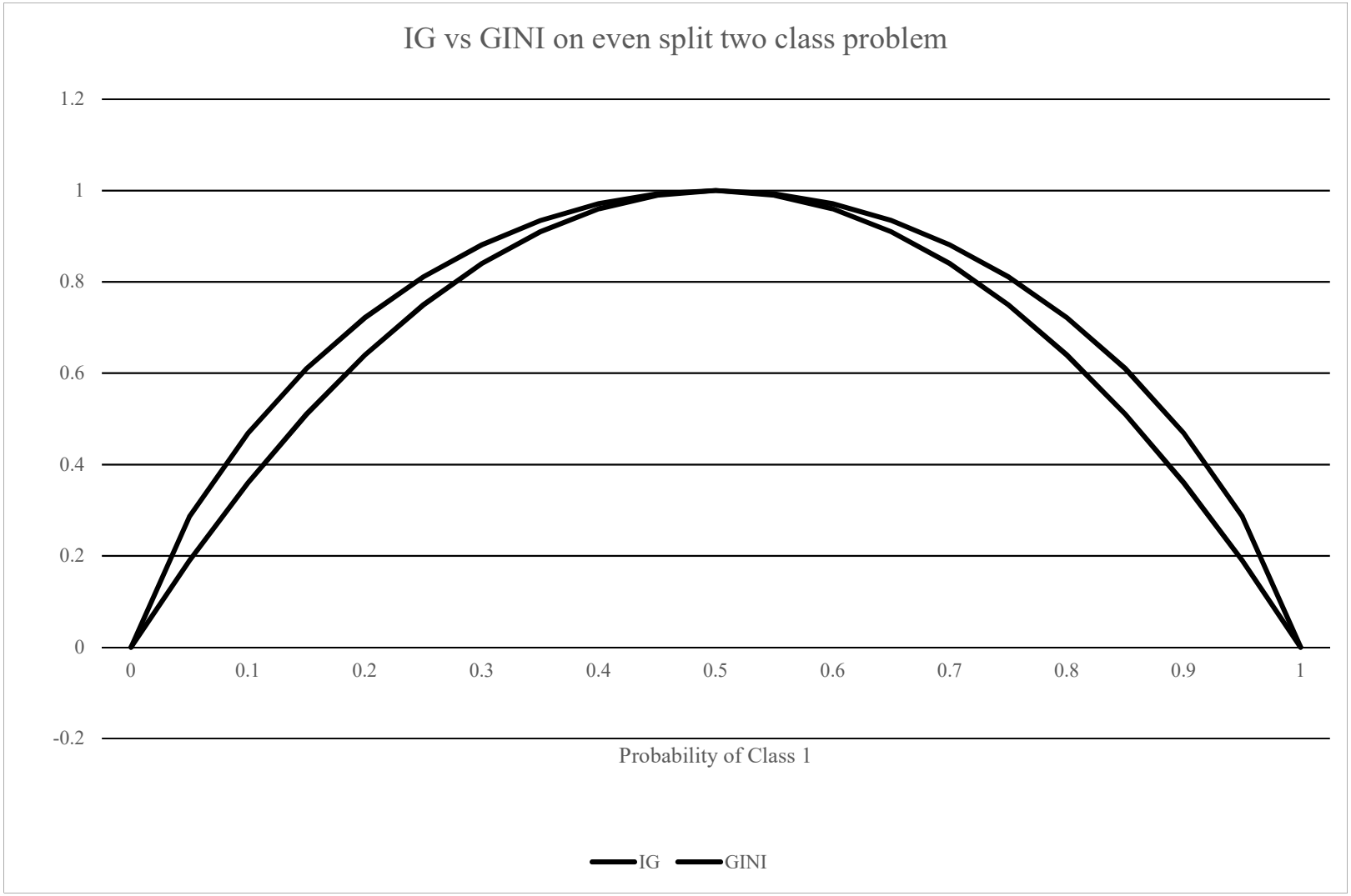
$$\text{I}(X) = 1 - \sum_{i=1}^{c} p(X = i)^2$$

$$\text{Gini}(X, A) = I(X) - \sum_{Y \in V} \frac{|Y|}{|X|} I(Y)$$

- The difference is mostly historical.
- Quinlan is a computer scientist, so naturally leaned towards information theory.
- Breiman was a statistician, and Gini comes from that area

Information Gain or Gini?

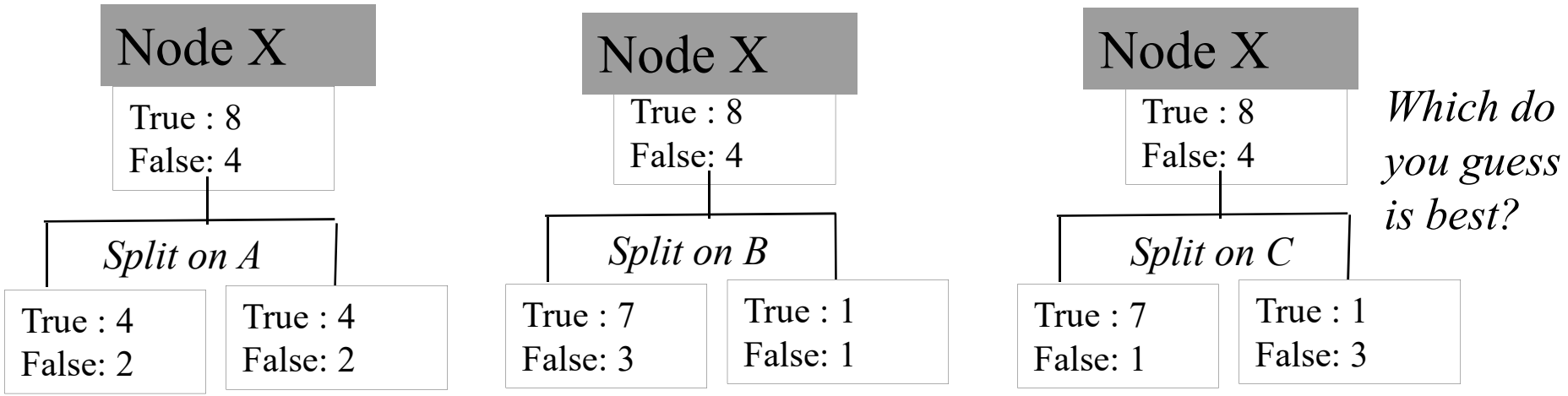
There is no obvious advantage of one over the other, although there has been plenty of discussion in the literature.



3. Chi Squared Statistic

Chi-Squared is a splitting measure that is quite different to IG and Gini. Consider again this situation. The distribution at the root is

$$\mathrm{p}(true) = 2/3 \qquad \mathrm{p}(false) = 1/3$$



Chi Squared is an alternative way of assessing an attribute based on the **chi-squared hypothesis test of goodness of fit**

Chi Squared Attribute Measure

- Information gain and Gini are just two possible measures of a quality of a split
- An alternative is the chi-squared statistic, that is used to answer the question *"is the class distribution significantly different after splitting"*

Root Node	Positive	Negative	Total
Count	50	150	200
Probability	50/200=0.25	150/200=0.75	1

Suppose an attribute splits it into two set of cases size 80 and 120

Root Node	Positive	Negative	Total
AttValue1			80
AttValue2			120
Total	50	150	200
Probability	50/200=0.25	150/200=0.75	1

Expected Values for Useless Attributes

- If the attribute does not help predict the class, we would expect the distribution of positive and negative to be about the same as the overall distribution

	(expected values)		
AttValue1	(80*0.25=20)	(80*0.75=60)	80
AttValue2	(120*0.25=30)	(120*0.75=90)	120
Total	50	150	200
Global Prob	0.25	0.75	

- If the observed values are very different to the expected values, then the attribute is good at discriminating between class values.

Contingency Tables

- A “good” contingency table

	Observed values and (expected values)		
AttValue1	45 (20)	35 (60)	80
AttValue2	5 (30)	115 (90)	120
Total	50	150	200
Global Prob	0.5	0.5	

- A “bad” contingency table

	Observed values and (expected values)		
AttValue1	18 (20)	62 (60)	80
AttValue2	32 (30)	88 (90)	120
Total	50	150	200
Global Prob	0.5	0.5	

Chi-Squared Statistic

	Observed values and (expected values)		
AttValue1	o_{11} (e_{11})	o_{12} (e_{12})	$o_{11} + o_{12}$
AttValue2	o_{21} (e_{21})	o_{22} (e_{22})	$o_{21} + o_{22}$
Total	$e_{11} + e_{21}$	$e_{12} + e_{22}$	
Global Prob			

Given a contingency table with *r* rows and *c* columns, let

e_{ij} is the expected value for cell *ij*

o_{ij} Is the observed value for cell *ij*

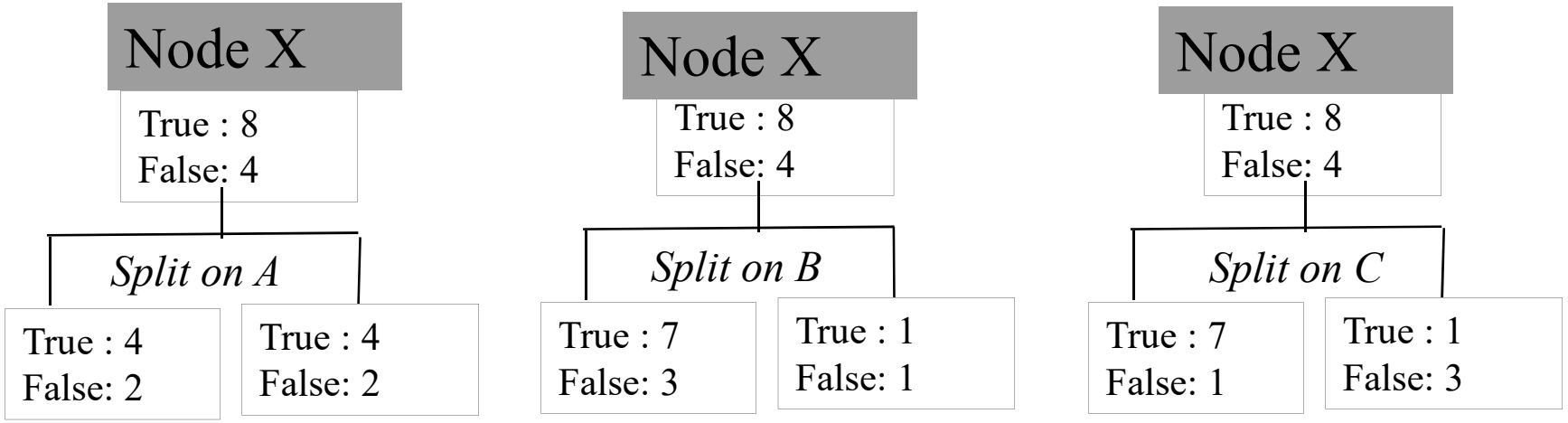
Chi-Squared Statistic is $$\chi^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(o_{ij} - e_{ij})^2}{e_{ij}}$$

Chi Squared

Suppose we need to choose between three attributes that produce the following splits of the train data

The distribution at the root is

$$\mathrm{p}(true) = 2/3 \qquad \mathrm{p}(false) = 1/3$$



We wish to choose the split that has the highest chi-squared statistic

Chi Squared Attribute A and B

	Observed values and (expected values)		
A1	4 (4)	2 (2)	6
A2	4 (4)	2 (2)	6
Total	8	4	12
Global Prob	2/3	1/3	

$$\chi^2 = 0$$

	Observed values and (expected values)		
B1	7 (10*2/3=6.67)	3 (10*1/3=3.33)	10
B2	1 (1.33)	1 (0.67)	2
Total	8	4	12
Global Prob	2/3	1/3	

$$\chi^2 = \frac{(7-6.67)^2}{6.67} + \frac{(3-3.33)^2}{3.33} + \frac{(1-0.67)^2}{0.67} \frac{(1-1.33)^2}{1.33} = 0.3$$

Information Gain Attribute C

	Observed values and (expected values)		
C1	7 (8*2/3=5.33)	1 (2.67)	8
C2	1 (2.67)	3 (1.33)	4
Total	8	4	12
Global Prob	2/3	1/3	

$$\chi^2 = \frac{(7-5.33)^2}{5.33} + \frac{(1-2.67)^2}{2.67} + \frac{(1-2.67)^2}{2.67}\frac{(1-1.33)^2}{1.33} = 5.333$$

Attribute A: chi squared 0
Attribute B: chi squared 0.33
Attribute C: chi squared 5.33

Choose C

Play Golf Example with Chi Squared

Root Node X	
Yes: 9 No : 5	P(True) : 9/14 P(False): 5/14

	Observed values and (expected values)		
AttValue1			
AttValue2			
Total	9	5	
Global Prob	9/14	5/14	

We need to assess which of the four attributes gives us highest chi-squared

Input Variables				Class
Outlook	Temp	Humidity	Windy	Play Golf
sunny	high (85)	high	0 (Weak)	0 (No)
sunny	high (80)	high	1 (Strong)	0
overcast	high (83)	high	0	1 (Yes)
rain	low (70)	high	0	1
rain	low (68)	high	0	1
rain	low (65)	low	1	0
overcast	low (64)	low	1	1
sunny	low (72)	high	0	0
sunny	low (69)	low	0	1
rain	low (75)	high	0	1
sunny	low (75)	low	1	1
overcast	low (72)	high	1	1
overcast	high (81)	low	0	1
rain	low (71)	high	1	0

Chi Squared Outlook

	Observed values and (expected values)		
Sunny	2 (5*9/14=3.21)	3 (5*5/14=1.79)	5
Overcast	4 (5*9/14=2.57)	0 (4*9/14=1.43)	4
Rain	3 (5*9/14=3.21)	2 (5*5/14=1.79)	5
Total	9	5	14
Global Prob	9/14	5/14	

$$\chi^2 = \frac{(2-3.21)^2}{3.21} + \frac{(3-1.79)^2}{1.79} + \frac{(4-2.57)^2}{2.57} + \frac{(0-1.43)^2}{1.43} + \frac{(3-3.21)^2}{3.21} + \frac{(2-1.79)^2}{1.79}$$

=3.54

TEMP	Observed values and (expected values)		
High	2 (5*9/14=2.57)	2 (4*5/14=1.43)	4
Low	7 (10*9/14=6.43)	3 (10*5/14=3.57)	10
Total	9	5	14
Global Prob	9/14	5/14	

$$\chi^2 = \frac{(2-2.57)^2}{2.57} + \frac{(2-1.43)^2}{1.43} + \frac{(7-6.43)^2}{6.43} + \frac{(3-3.57)^2}{3.57} = 0.5$$

HUMIDITY	Observed values and (expected values)		
High	5 (9*9/14=5.78)	4 (9*5/14=3.22)	9
Low	4 (5*9/14=2.57)	1 (5*5/14=1.79)	5
Total	9	5	14
Global Prob	9/14	5/14	

$$\chi^2 = \frac{(5-5.78)^2}{5.78} + \frac{(4-3.22)^2}{3.22} + \frac{(4-2.57)^2}{2.57} + \frac{(1-1.79)^2}{1.79} = 0.44$$

Windy	Observed values and (expected values)		
WEAK	6 (8*9/14=5.14)	2 (8*5/14=2.86)	8
STRONG	3 (6*9/14=3.86)	3 (6*5/14=2.14)	6
Total	2	3	14
Global Prob	9/14	5/14	

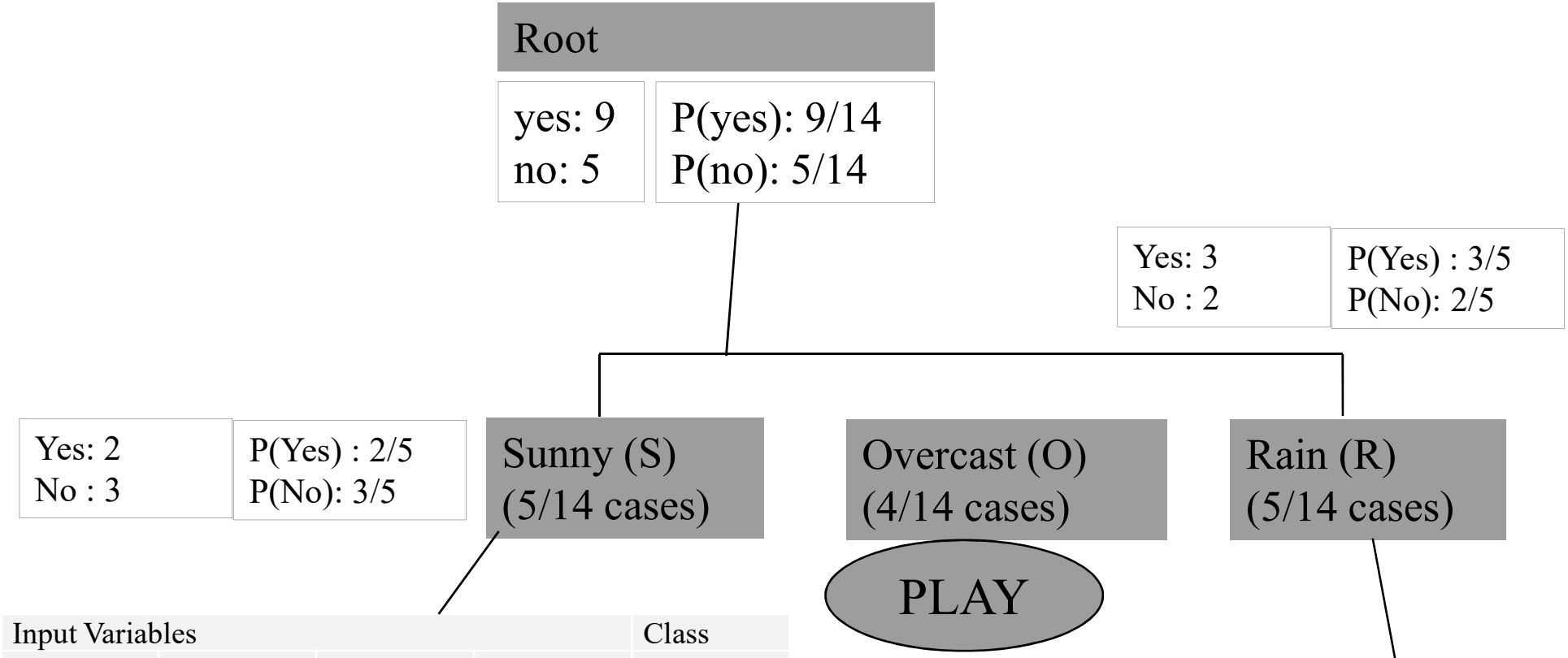
$$\chi^2 = \frac{(6-5.14)^2}{5.14} + \frac{(2-2.86)^2}{2.86} + \frac{(3-3.86)^2}{3.86} + \frac{(3-2.14)^2}{2.14} = 0.933$$

$$\text{chi}(X, outlook) = 3.54$$
$$\text{chi}(X, windy) = 0.933$$
$$\text{chi}(X, temp) = 0.5$$
$$\text{chi}(X, humidity) = 0.44$$

Choose Outlook

$$\text{Gain}(X, outlook) = 0.24675$$
$$\text{Gain}(X, windy) = 0.0481$$
$$\text{Gain}(X, humidity) = 0.0453$$
$$\text{Gain}(X, temp) = 0.025$$

Choose Outlook



Input Variables				Class
Outlook	Temp	Humidity	Windy	Play Golf
sunny	high (85)	high	0 (Weak)	0 (No)
sunny	high (80)	high	1 (Strong)	0
sunny	low (72)	high	0	0
sunny	low (69)	low	0	1
sunny	low (75)	low	1	1

Input Variables				Class
Outlook	Temp	Humidity	Windy	Play Golf
rain	low (70)	high	0	1
rain	low (68)	high	0	1
rain	low (65)	low	1	0
rain	low (75)	high	0	1
rain	low (71)	high	1	0

TEMP/SUNNY	Observed values and (expected values)		
High	0 (2*2/5=0.8)	2 (2*3/5=1.2)	2
Low	2 (3*2/5=1.2)	1 (3*3/5=1.8)	3
Total	2	3	5
Global Prob	2/5	3/5	

$$\chi^2 = \frac{(0-0.8)^2}{0.8} + \frac{(2-1.2)^2}{1.2} + \frac{(2-1.2)^2}{1.2} + \frac{(1-1.8)^2}{1.8} = 2.22$$

HUMIDITY/ SUNNY	Observed values and (expected values)		
High	0 (3*2/5=1.2)	3 (3*3/5=1.8)	3
Low	2 (2*2/5=0.8)	0 (2*3/5=1.2)	2
Total	2	3	5
Global Prob	2/5	3/5	

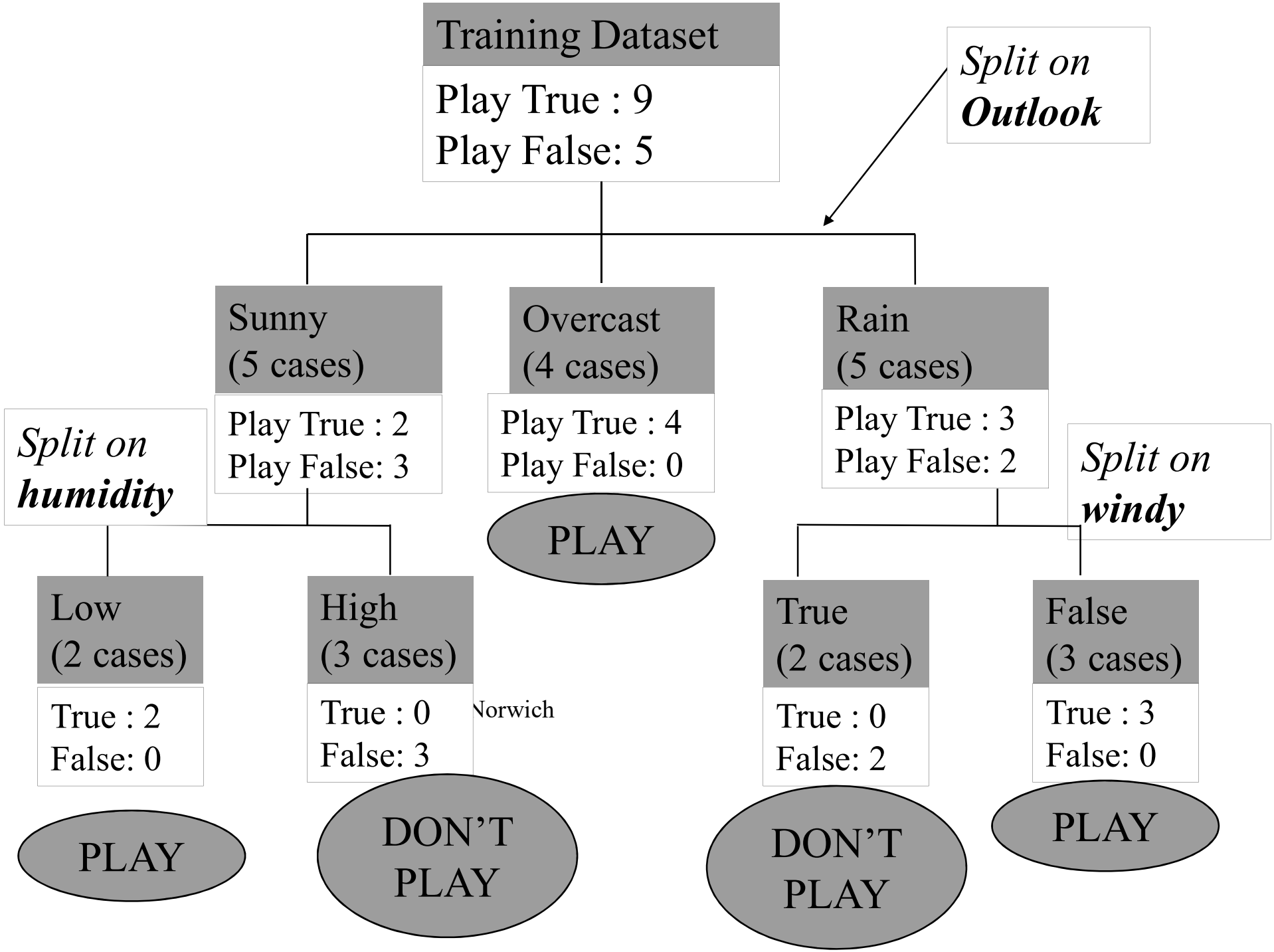
$$\chi^2 = \frac{(0-1.2)^2}{1.2} + \frac{(3-1.8)^2}{1.8} + \frac{(2-0.8)^2}{0.8} + \frac{(0-1.2)^2}{1.2} = 5$$

Windy	Observed values and (expected values)		
WEAK	1 (3*2/5=1.2)	2 (3*3/5=1.8)	3
STRONG	1 (2*2/5=0.8)	1 (2*3/5=1.2)	2
Total	2	3	5
Global Prob	2/5	3/5	

$$\chi^2 = \frac{(1-1.2)^2}{1.2} + \frac{(2-1.8)^2}{1.8} + \frac{(1-0.8)^2}{0.8} + \frac{(1-1.2)^2}{1.2} = 0.14$$

Choose humidity

$$\text{chi}(X, windy) = 0.14$$
$$\text{chi}(X, temp) = 2.22$$
$$\text{chi}(X, humidity) = 5$$


Training Dataset
Play True : 9
Play False: 5
Split on
Outlook
Sunny
(5 cases)
Play True : 2
Play False: 3
Overcast
(4 cases)
Play True : 4
Play False: 0
Rain
(5 cases)
Play True : 3
Play False: 2
Split on
humidity
Split on
windy
Low
(2 cases)
True : 2
False: 0
High
(3 cases)
True : 0
False: 3
True
(2 cases)
True : 0
False: 2
False
(3 cases)
True : 3
False: 0
PLAY
PLAY
DON'T PLAY
DON'T PLAY
PLAY

Decision Trees

Part 3: Decision Tree Algorithms

1. Design Issues with DTs
    1. Attribute Characteristics
    2. Overfitting and Pruning

Decision Trees

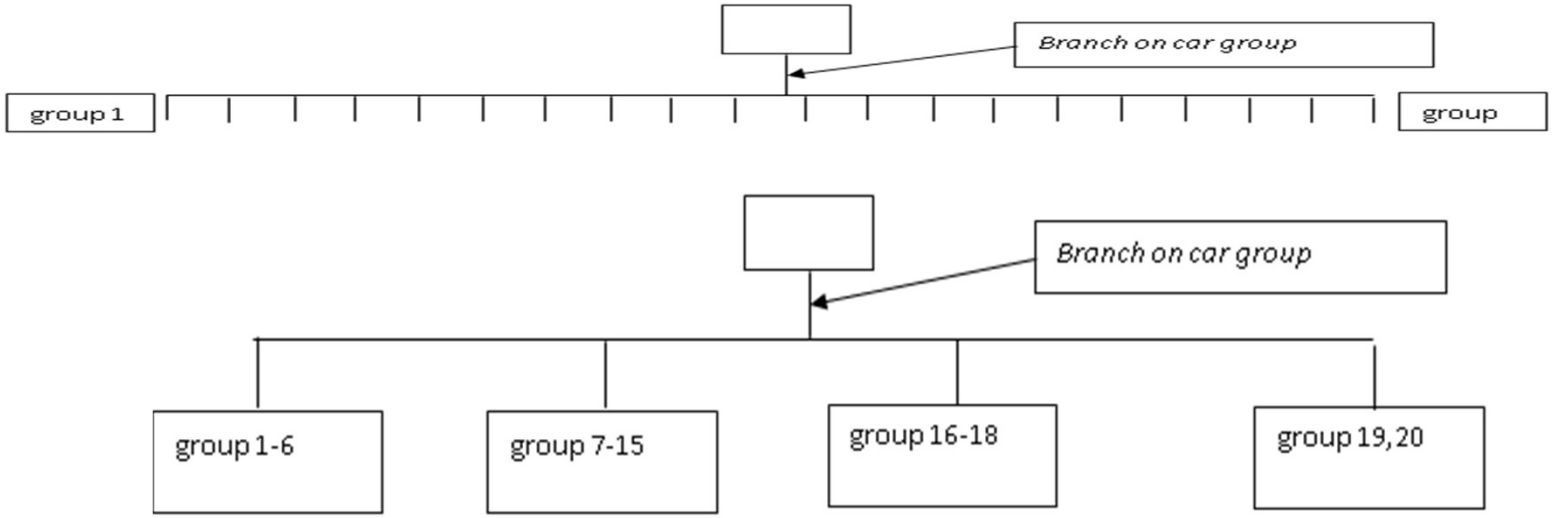
Part 3: Decision Tree Algorithms

1. ID3/C4.5/C5
2. AID/CHAID/KnowledgeSeeker
3. CART
4. Other variants

Discrete Attributes

Discrete attributes can have different number of values

1. Does this effect the split measure, i.e. do measures give undue preference with attributes to few/many values?
2. Do we want to produce a new node for each value, or somehow group them? Suppose we have an attribute "car group" for an insurance problem.



Continuous Attributes

- Suppose we have a continuous attribute, such as temperature. How do we form splits? We can either

1. Look for binary splits (e.g. Temp<=75)

If we do this we need to determine the best split point, and we also need to allow a continuous attribute to appear more than once in the tree

2. Discretise the data into bins (e.g. low and high) and continue as before

We can either do this before any construction, or do it on the fly as we go.

Overfitting

- Overfitting occurs when noise in the training data cause the algorithm to construct a too complex tree that will misclassify new cases.
- There are two approaches to stopping overfitting a tree:

1. Stop growing the tree using a statistical test of significance.
2. Allow the tree to overfit, then use an algorithm after it has finished to **"prune"** the tree (i.e. remove unnecessary branches)

Decision Trees

Part 3: Decision Tree Algorithms

1. ID3/C4.5/C5
2. AID/CHAID/KnowledgeSeeker
3. CART
4. Other variants

Algorithm Family 1:

The Iterative Dichotomiser 3
ID3

ID3 was the first of three decision tree algorithms invented by Ross Quinlan

Stopping Criteria	*Continue until node is all one class or all attributes have been used*
How to Split the Data	*Only works on discrete data. Node for each discrete attribute value*
Choosing an Attribute	*Choose the attribute with the maximum information gain*

Quinlan, J. R. 1986. *Induction of Decision Trees*. **Machine Learning**. 1, 1 (Mar. 1986), 81–106

ID3/**C4.5**/C5

Three algorithms all based on Information Gain

1. ID3*. Uses information gain, builds full tree, discrete variables only

2. C4.5**: Uses **information gain ratio** instead of information gain. Looks for **all binary splits** in continuous attributes. Performs **reduced error pruning** on a full tree.

2. C5/See5: A commercial product with several claimed improvements

*Quinlan, J. R. 1986. *Induction of Decision Trees*. **Machine Learning**. 1, 1 (Mar. 1986), 81–106
**Quinlan, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993

C4.5: Information Gain Ratio

Information Gain can bias towards attributes with larger number of values

$$\text{GainRatio}(X, A) = \frac{Gain(X,A)}{SplitInfo(X,A)}$$

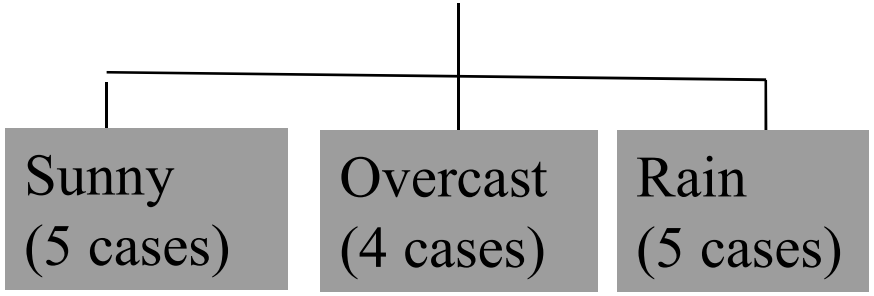
$$\text{SplitInfo}(X, A) = -\sum_{V_i \in V} \frac{|V_i|}{|X|} log\left(\frac{|V_i|}{|X|}\right)$$

$$V = \{A_1, A_2, \ldots, A_k\}$$

Where V is the set of nodes for each of the k possible values for attribute A.

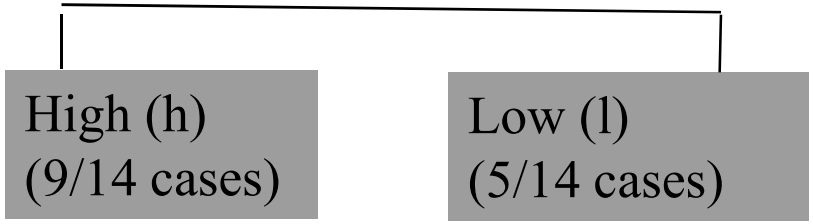
Favours attributes with fewer nodes

C4.5: Split Info Example

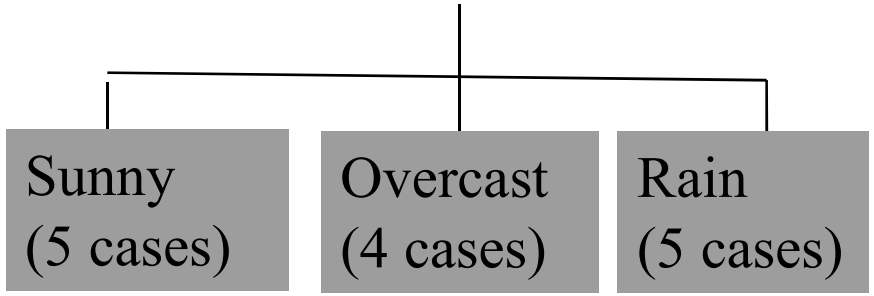


This attribute gets an advantage because it has more target values

$$\text{SplitInfo}(X,A) = \frac{5}{14} log\left(\frac{5}{14}\right) + \frac{4}{14} log\left(\frac{4}{14}\right) + \frac{5}{14} log\left(\frac{5}{14}\right) = 1.577$$



$$\text{SplitInfo}(X,A) = \frac{9}{14} log\left(\frac{9}{14}\right) + \frac{5}{14} log\left(\frac{5}{14}\right) = 0.94$$

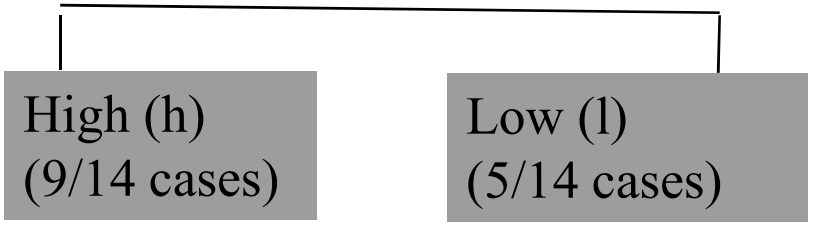


H(X) = 0.94

$\text{Gain}(X, A) = 0.24675$

$\text{SplitInfo}(X, A) = 1.577$

$\text{GainRatio}(X, A) = 0.24675/1.577 = 0.1565$



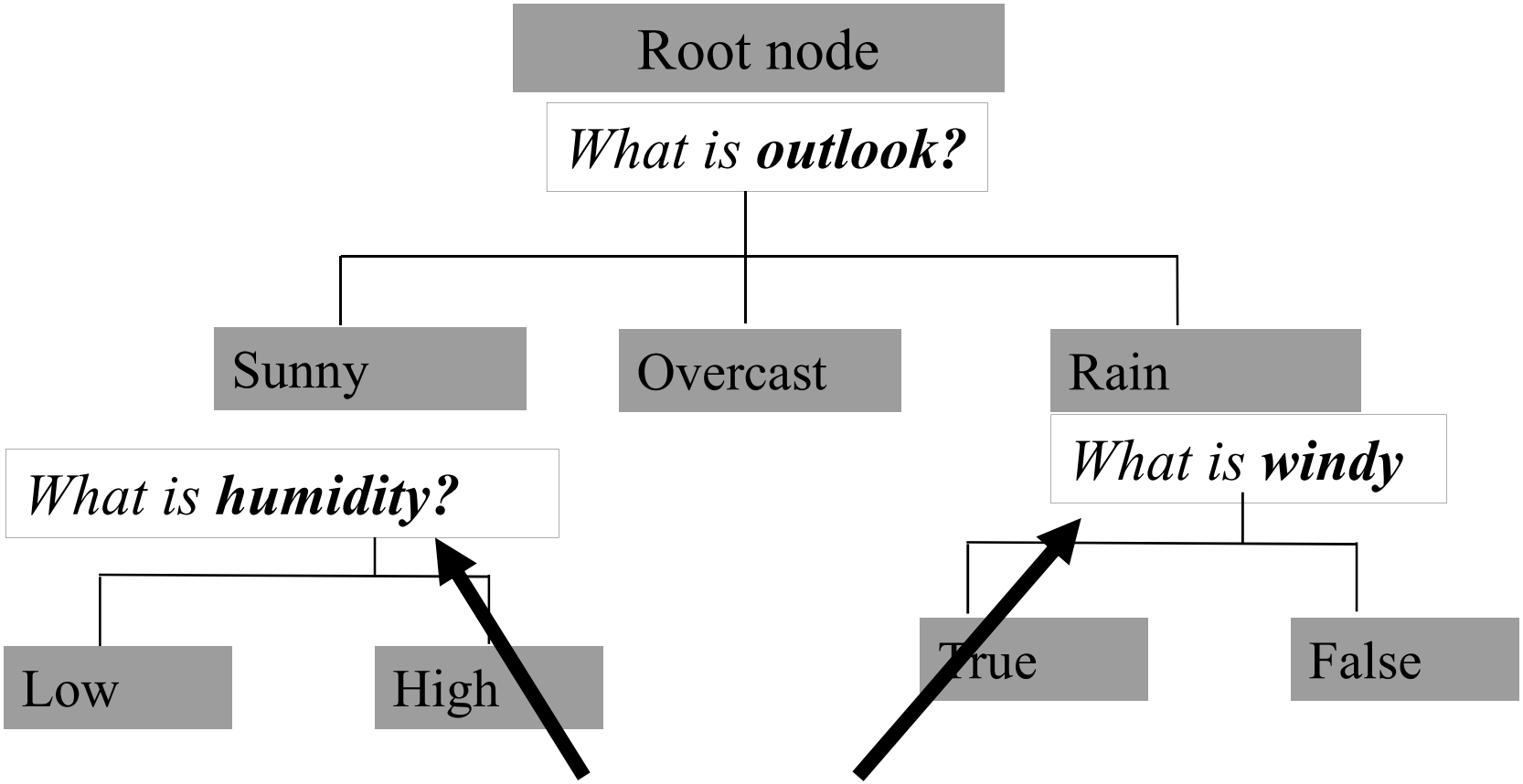
$\text{Gain}(X, A) = 0.025$

$\text{SplitInfo}(X, A) = 0.94$

$\text{GainRatio}(X, A) = 0.025/0.94 = 0.0266$

No difference in decision in this case, but the gap between the two attributes is less

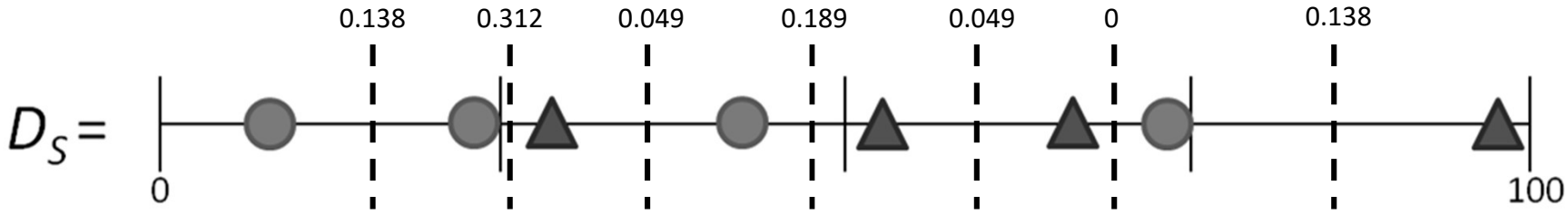
C4.5 Pruning: removing parts of the tree we don't need



Do we need these branches? If we remove them does it make any difference? ***See the evaluation lecture***

C4.5 Continuous Attributes

- C4.5 performs binary splits on continuous attributes, which can be selected multiple times in a tree
- It evaluates all possible split points and uses the one with the highest information gain
- This is fairly slow



the position of the best split point is used as the decision criteria for the decision tree

Algorithm Family 2: AID/CHAID

*AID: Automatic Interaction Detector** was a decision tree algorithm proposed in 1963.

CHAID****: Chi-Squared Automatic Interaction Detector** was proposed in 1980 by G. V. Kass in his PhD thesis. Key features

Stopping Criteria	**Stops early.** Only continues to split if a significant improvement occurs
How to Split the Data	**Discretises** real valued attributes into bins Groups values of discrete attributes if it improves performance
Choosing an Attribute	Uses **Chi-Squared** to assess attributes

*Morgan, J. and Sonquist, J. Automatic Interaction Detection (AID) Techniques Journal of the American Statistical Association, Vol. 58, 1963.

**Kass, Gordon V.; *An Exploratory Technique for Investigating Large Quantities of Categorical Data*, Applied Statistics, Vol. 29, No. 2 (1980), pp. 119–127

Algorithm Family 3: CART

Classification and Regression Tree.

Stopping Criteria

- Stops when a minimum number of cases is in a node
- Uses post building **pruning**

How to Split the Data

- **Binary Trees only**: Evaluates all splits in continuous attributes, groups together discrete values for attributes with more than two

Choosing an Attribute

- Uses the **Gini index** to assess splits

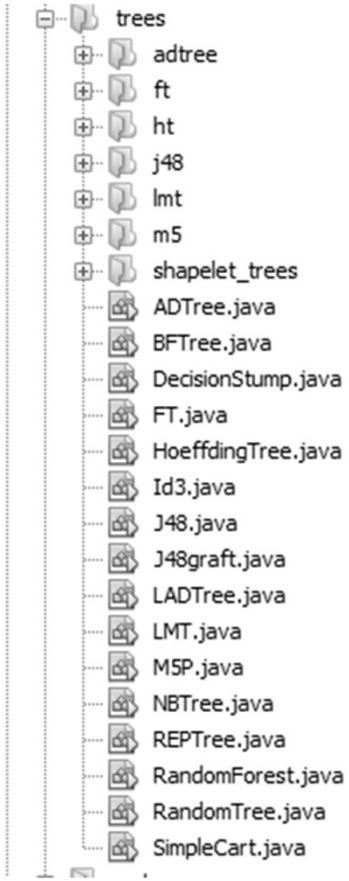
- Embedded in commercial software from Salford Systems

Breiman, L, Friedman, J, Stone, C and Olshen, R, *Classification and Regression Trees*. **Wadsworth Publishing, 1984**

Other Decision Trees and Implementations

Weka

weka.classifiers.trees



Id3/J48/J48Graft: implementations of ID3 and C4.5

SimpleCart: implementation of CART

ADTree: alternating decision tree. Reweights instances at each node

DecisionStump: Base classifier for boosting ensembles

Random Tree

Base Classifier for ensemble Random Forest (see ensembles lecture). Considers a random subset of attributes at each node

Decision Trees Implementations

Scikit-learn

sk-learn has a single, configurable Decision Tree

`sklearn.tree`**.DecisionTreeClassifier**

```
class DecisionTreeClassifier(BaseDecisionTree, ClassifierMixin):
    """A decision tree classifier.
```

You set it up via the constructor

```
def __init__(self,
             criterion="gini",
             splitter="best",
             max_depth=None,
             min_samples_split=2,
             min_samples_leaf=1,
             min_weight_fraction_leaf=0.,
             max_features=None,
             random_state=None,
             max_leaf_nodes=None,
             min_impurity_decrease=0.,
             min_impurity_split=None,
             class_weight=None,
             presort=False):
```

I'm not sure about alternative means of dealing with continuous variables, different stopping conditions or pruning

And finally….

Decision Trees Pros:

1. Fast to build and use
2. Easy to understand how classification occurs
3. A good “base” classifier for ensembles.

Decision Trees Cons:

1. Not competitive in terms of accuracy with more modern approaches (see ensembles lecture)
2. Does not produce good probability estimates