Lecture 6

- Introduction to decision trees
- Fitting a decision tree by hand
- Fitting a decision tree in R

Presenter: Dr Heshan Kumarage

Week-by-week outline

Week Starting	Seminar	Topic	App Ses	A1	A2	Q/P	A3	Due Date
2/3/2026	1	Introduction to Data Science, R, review of basic statistics	-					
9/3/2026	2	Data visualisation	S1					
16/3/2026	3	Data manipulation	S2					
23/3/2026	4	Regression modelling	S3					
30/3/2026	5	Clustering	S4					
6/4/2026	-	Mid-semester Break						
13/4/2026	6	Classification using decision trees	S5					17/4/2026
20/4/2026	7	Improving and evaluating classifiers. Naïve Bayes classification	S6					
27/4/2026	8	Ensemble methods, Artificial Neural Networks	S7					
4/5/2026	9	Network analysis	S8					
11/5/2026	10	Introduction to text analysis	S9					15/5/2026
18/5/2026	11	Text analysis applications	Quiz/Prac					22/5/2026
25/5/2026	12	Text Network Analysis, Review of the unit, Assignment 3	S10,11,12					
1/6/2026		SWOT VAC	-					
8/6/2026		EXAM PERIOD	-					12/6/2026

Assignment 1

MONASH University

Faculty of Information Technology

FIT3152 Data analytics – 2026: Assignment 1

Your task	• Analyse the country level predictors of confidence in **social organisations and how these change over time** using data from the World Values Survey. • This is an individual assignment.
Value	• This assignment is worth **25%** of your total marks for the unit. • It has **40** marks in total.
Suggested Length	• 8 – 10 A4 pages, approximately 2,000 words (for your report) + extra pages as appendix for your R script and report on how Generative AI used, if required. • Font size 11 or 12pt, single spacing.

Assignment 1

Due Date	**11.55pm Friday 17th April 2026**
Submission	• Submit a single PDF file **and** single video file on Moodle. • Note that submission of a video report is a hurdle requirement. • Use the naming convention: *FirstnameSecondnameID.{pdf, mp4, mov etc.}* • Turnitin will be used for similarity checking of all written submissions.
Generative AI Use	• In this assessment, you can use generative artificial intelligence (AI) in order to search for R functions and examples to perform tasks that you specify only. Any use of generative AI must be appropriately acknowledged (see Learn HQ).
Late Penalties	• 5% (**2** mark) deduction per calendar day for up to one week. • Submissions more than 7 calendar days after the due date will receive a mark of zero (0) and no assessment feedback will be provided.

Outline

In this section we'll cover:

- Introduction to machine learning and classification
- Some examples of decision trees
- Applying a decision tree classifier

Machine Learning

We can think of Machine Learning (ML) as the automated (statistical) learning of a concept from labelled sample data.

- For example, spam filtering with an algorithm that takes some examples of spam and makes a rule to predict whether an email should go to the spam folder.

How can a model "learn" a concept?

- Descriptive: captures the "behaviour" of the training data;
- Predictive: generalizes to unseen data;
- Explanatory: describes the concept to be learned.

A typical application of ML is classification.

Classification

- Using a collection of observations containing a set of *attributes* where one of the attributes is the *class*, find a model to predict class as a function of the other attributes.
- Goal: previously unseen records should be assigned a class as accurately as possible.
- Data is usually divided into a *training set* to build the model, and a *test set* used to validate (test the quality of prediction) of the model.

Classification example – tax return

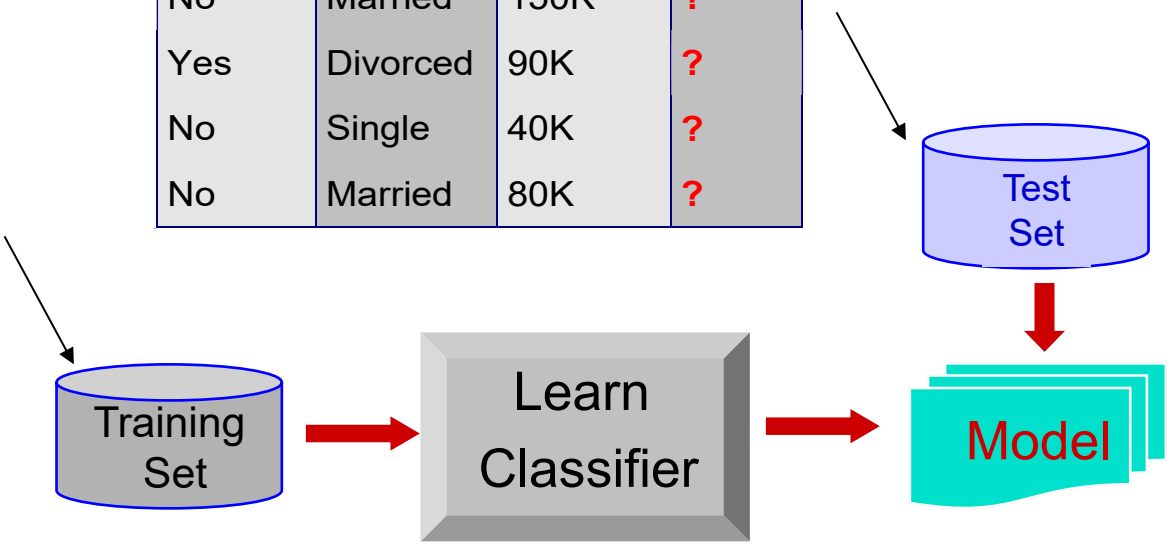
- Given the following data about people who submitted a tax return, we want to automatically create a model that will classify people into cheats or non-cheats.
- We then want to be able to Use the model to classify 'new' people into cheats or non-cheats.

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

Classification example – tax return

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

Refund	Marital Status	Taxable Income	Cheat
No	Single	75K	?
Yes	Married	50K	?
No	Married	150K	?
Yes	Divorced	90K	?
No	Single	40K	?
No	Married	80K	?



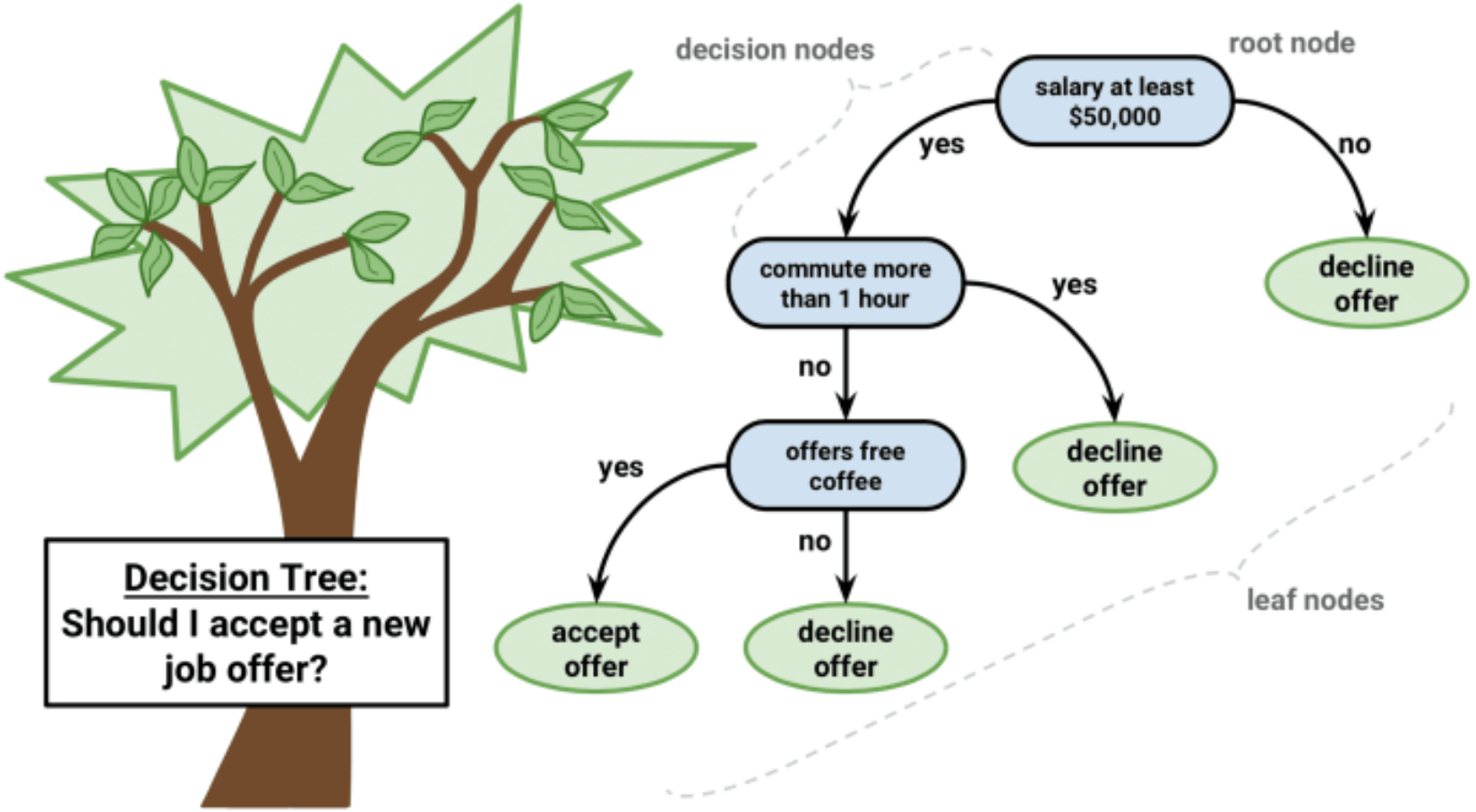
Classification

Machine Learning classifiers include:

- Decision Tree based methods
- Naïve Bayes and Bayesian Belief Networks
- Ensemble methods
- Rule-based methods
- Support Vector Machines
- Artificial Neural Networks
- Convolutional Neural Networks

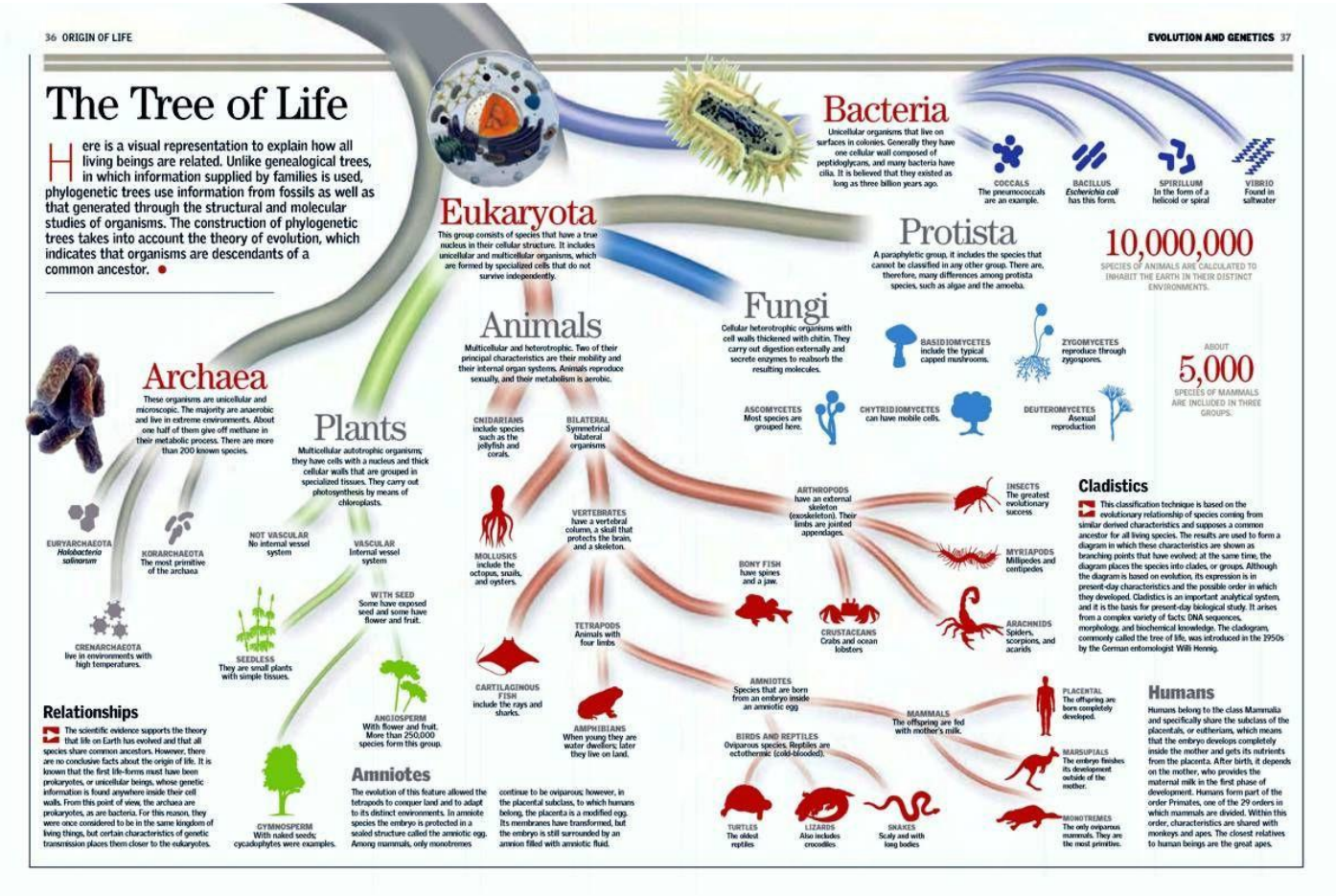
In each of these cases the model "learns" the classification rules from the data.

Decision tree: should I accept a job?



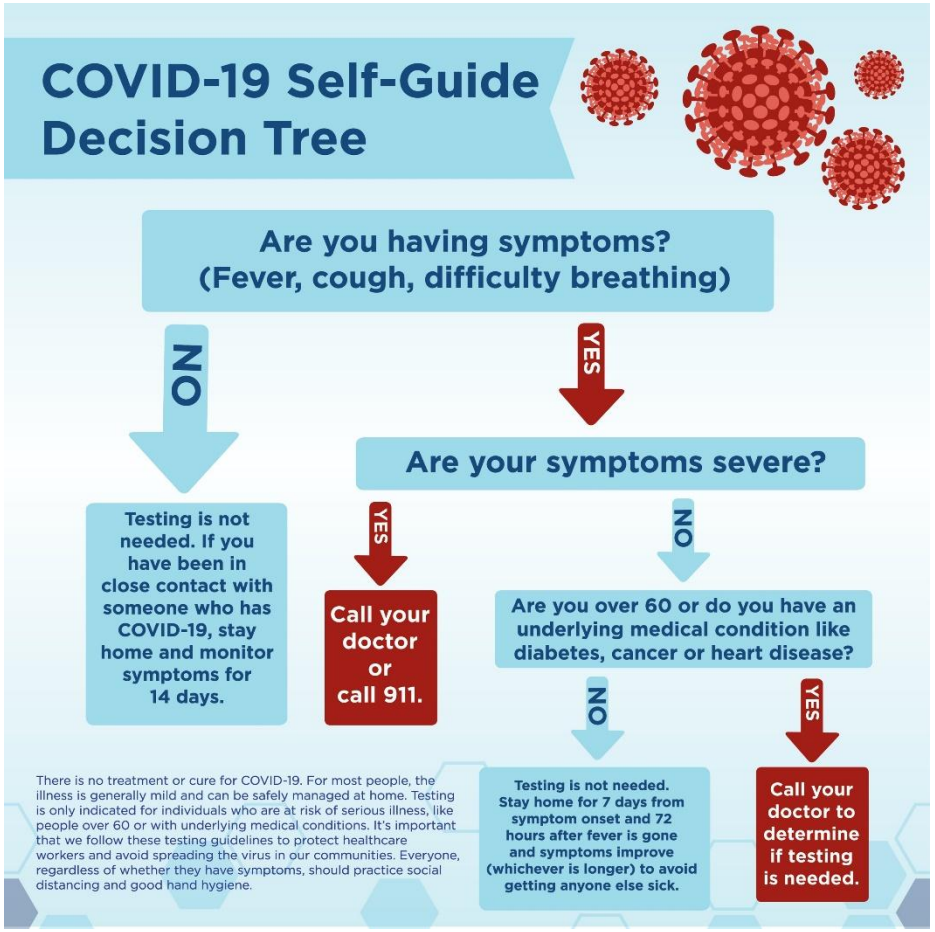
towardsdatascience.com/

Classification tree: life forms



medium.com/

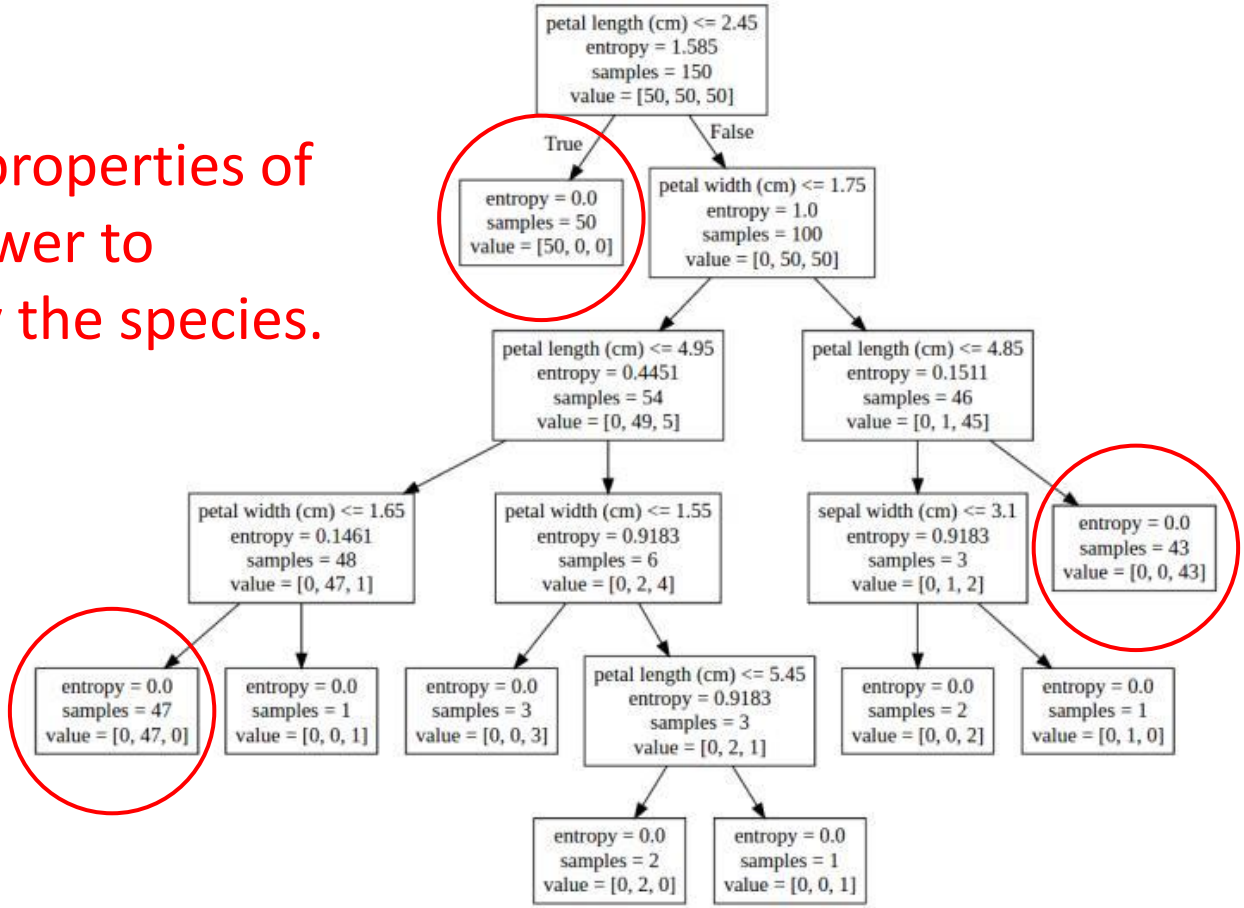
COVID-19 Self-diagnosis



holzer.org/

Iris species classification

Using properties of the flower to classify the species.



kdnuggets.com/

Decision Trees

Decision trees are one of the most widely used and practical methods of machine learning:

- Model trained on existing labelled data
- Can be used to classify new instances
- Can be used to profile existing data
- Robust to noise and missing values
- Each tree can be viewed as a sequence of "if – then – else" statements. This readability is highly desirable.
- *We can construct simple decision trees by hand!*

Decision Trees

Terminology

- *Instance*: single row in a data set. (each observation).
- *Attribute*: an aspect of an instance. Sometimes called a feature variable.
- *Value*: the category that an attribute can take.
- *Class*: the thing to be learned. Sometimes called the target attribute.
- It is usual to have several decision attributes and one class.

Decision Trees

A decision tree consists of:

- Leaf nodes (of each class) and non-leaf nodes, corresponding to the decision attributes,
- Branches – corresponding to the values of the decision attributes having either binary or multi-way splits.
- To classify an instance, each decision node (starting from the root) compares an attribute of the instance with a specific attribute value (or range) and takes the corresponding branch.
- A path from the root to a leaf node gives the class of the object.

Decision Trees

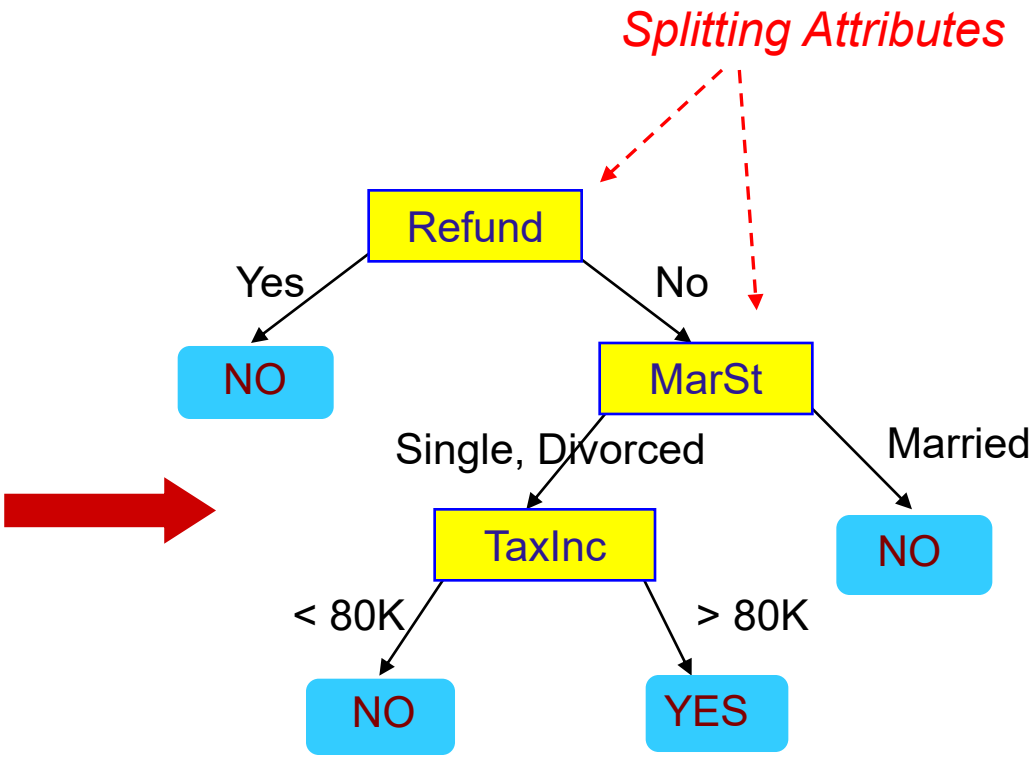
Further considerations:

- Type of decision tree:
  - > Classification Trees (categorical – nominal attributes)
  - > Regression Trees (numerical – continuous attributes)
- How do we specify the splitting conditions?
- How do we evaluate the decision tree model?

Classification example – tax return

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

Training Data

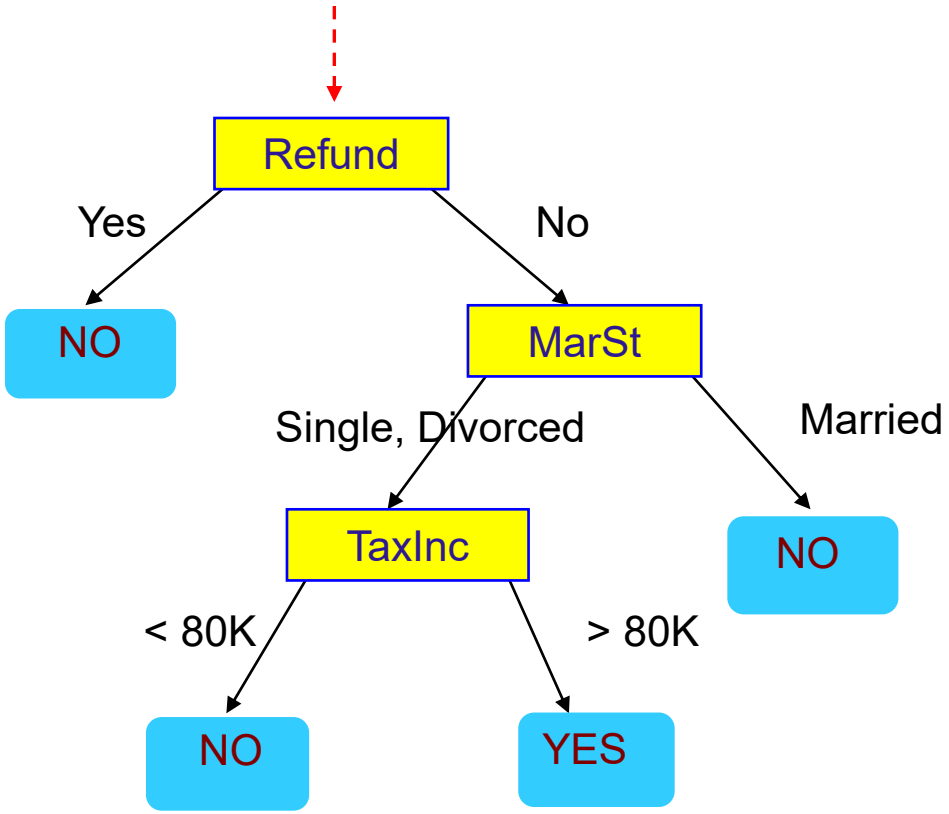


Model: Decision Tree

(One of many possible trees)

Apply Model to Test Data

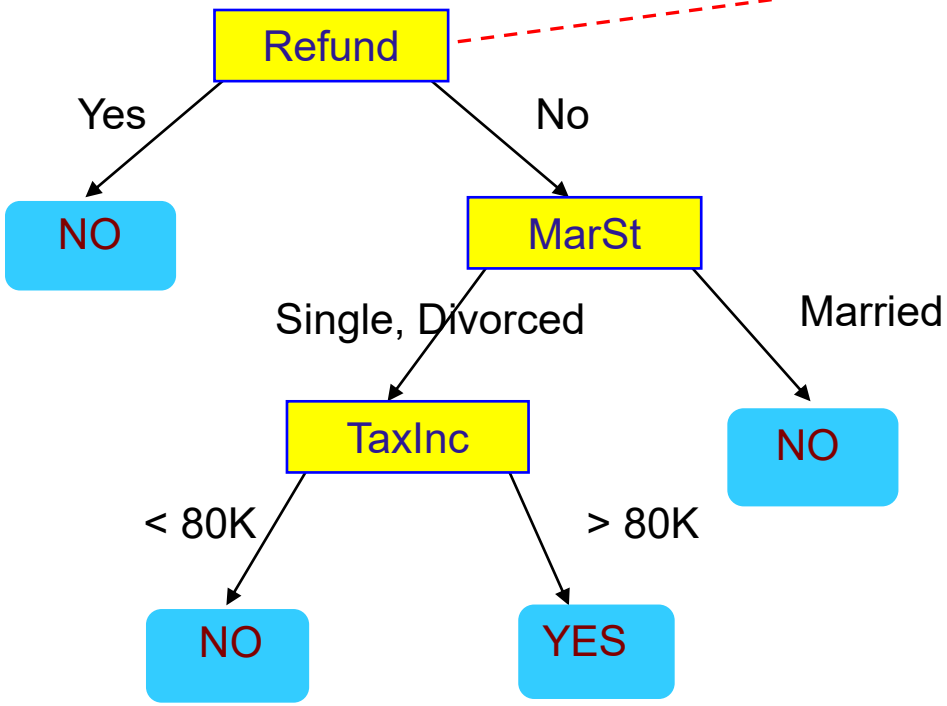
Start from the root of tree.



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

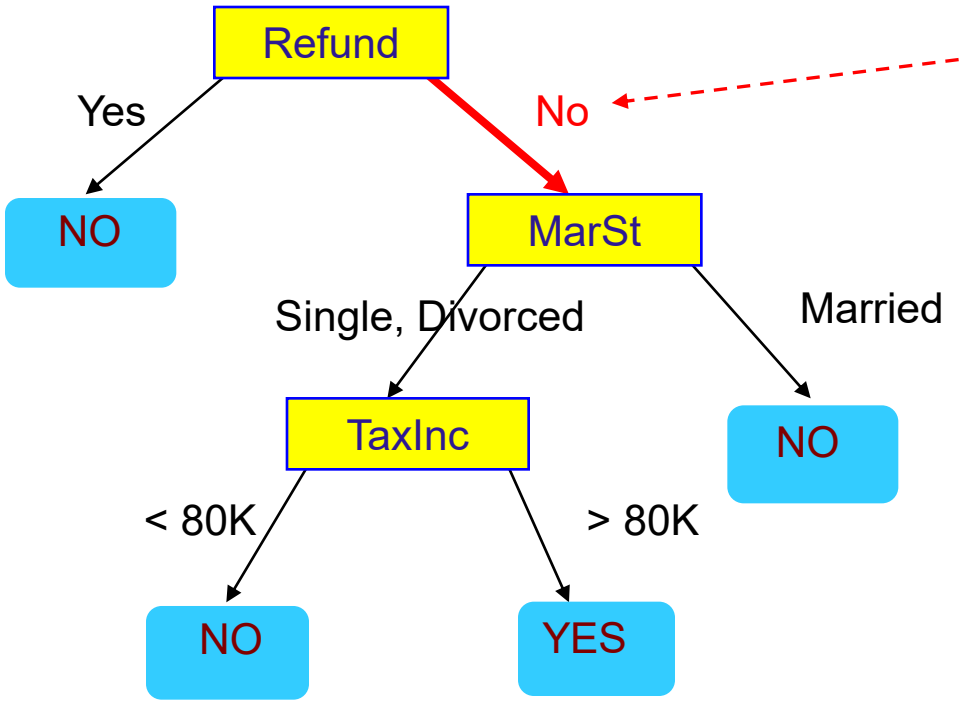
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



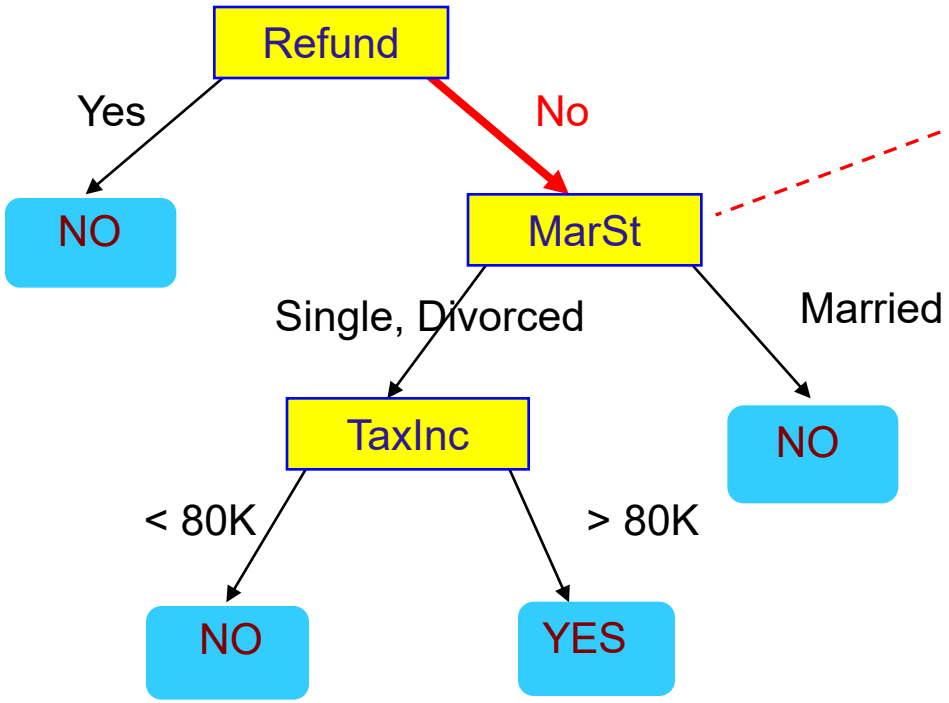
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

Test Data

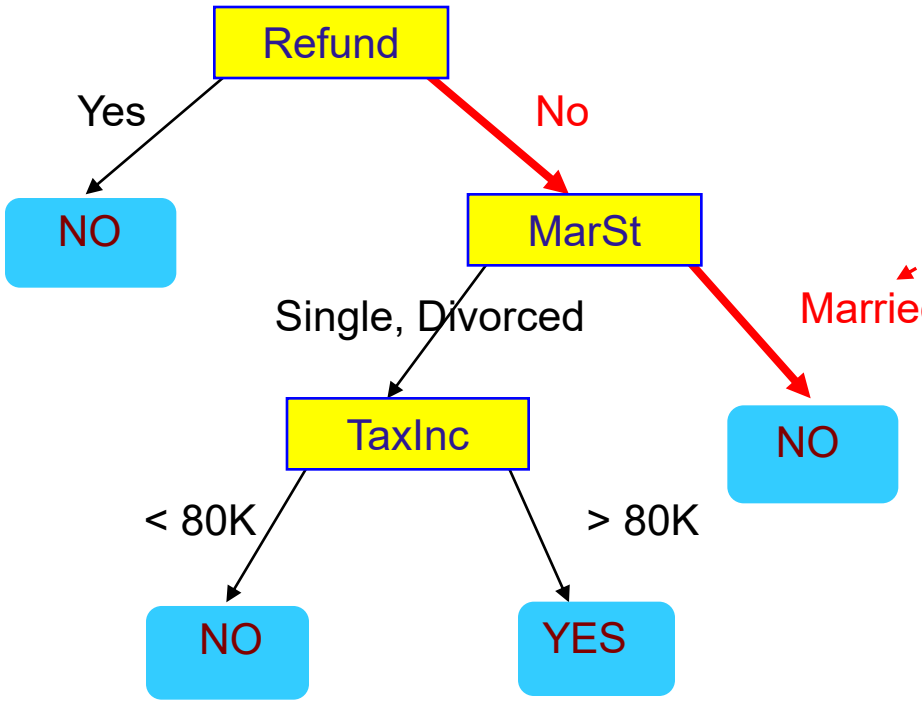
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



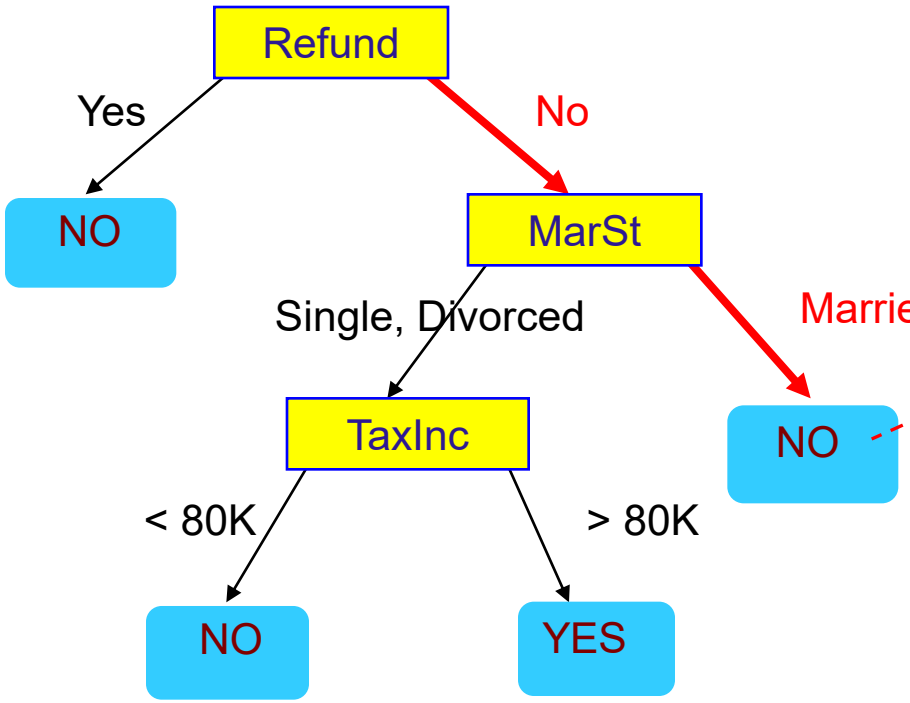
Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Assign Cheat to “No”

Fitting a decision tree by hand

In this section we'll cover:

- A specific decision tree algorithm: ID3
- Fitting a decision tree using ID3
- Entropy and information gain
- Model performance evaluation

Building a decision tree

Building a decision tree requires answering:

- Which attribute should be tested at a node?
- When should a node be declared a leaf?
- What if a tree becomes too large?
- How to handle missing values?
- Should the properties be restricted to binary-valued or allowed to be multi-valued?
- Answering these questions leads to different variants of decision trees: ID3, C4.5, C5, CART, etc.

Top-down induction: ID3

The algorithm (*Iterative Dichotomiser 3*):

- At each step, determine the "best" decision attribute, *A*.
- Assign *A* as decision attribute for node.
- For each value of *A* create new descendant.
- Sort training examples according to the attribute value.
- If all training examples are homogenous (i.e., perfectly classified), stop, else iterate over new leaf nodes.

For this algorithm we assume class attribute is categorical.

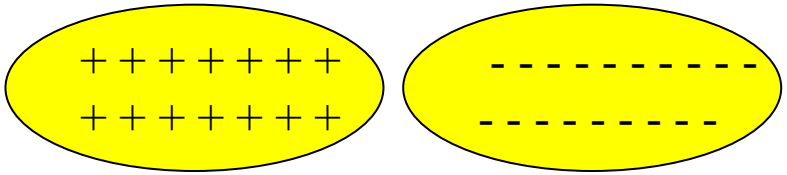
en.wikipedia.org/

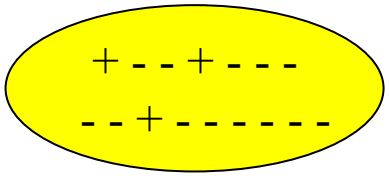
Which attribute to split on?

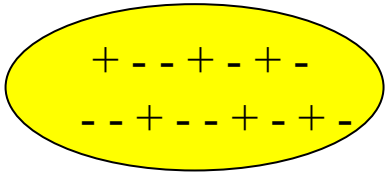
- At each stage of the process, we try to find the 'best' attribute and split to partition the data.
- At each split the goal is to increase the *homogeneity* of the resulting datasets with respect to the *class* or *target* variable (which we are trying to classify).
- That split may not be the best overall – but once it is made, we stay with it for the rest of the tree.
- This is generally called a *greedy* approach and may not result in the best overall decision tree.

Homogeneity

Suppose we have a binary target attribute with values '+' and '–'.

- These two sets are homogeneous 

- This one is not 

- This one less so 

Information gain

Which attribute to choose for splitting?

- A statistical property called *information gain* measures how well a given attribute separates the training examples into homogeneous groups according to target classification.
- ID3 uses information gain as the splitting criteria for building a tree and chooses the attribute which provides the greatest information gain.
- Information gain is determined using a measure from Information Theory called *Entropy*.

Entropy

In Thermodynamics:

- It gives a measure of the amount of chaos present in a system (or a measure of the disorder in a system).

In Information Theory:

- Entropy measures the uncertainty in a random variable or message or indicates how much information (or impurity) there is in an event.
- In general, the more uncertain or random the event is, the more information it will contain.

Entropy cont…

See Wikipedia:

https://en.wikipedia.org/wiki/Entropy_(information_theory)

- In information theory, systems are modeled by a transmitter, channel, and receiver… The receiver attempts to infer which message was sent. In this context, entropy (more specifically, Shannon entropy) is the expected value (average) of the information contained in each message. 'Messages' can be modeled by any flow of information…

Calculating entropy

- For a two-class problem: C_1 and C_2, where the probability of belonging to each class is P_{C_1} and P_{C_2}:
- $Entropy(S) = -P_{C_1} \cdot log_2(P_{C_1}) - P_{C_2} \cdot log_2(P_{C_2})$

- For a multi-class problem having n classes:
- $Entropy(S) = -\sum_{i=1}^{n} P_{C_i} \cdot log_2(P_{C_i})$

- The units of entropy when log_2 is used are "Shannons."

Calculating entropy

Suppose S is a collection of 14 examples, 9 positive and 5 negative → [9+ ,5-]

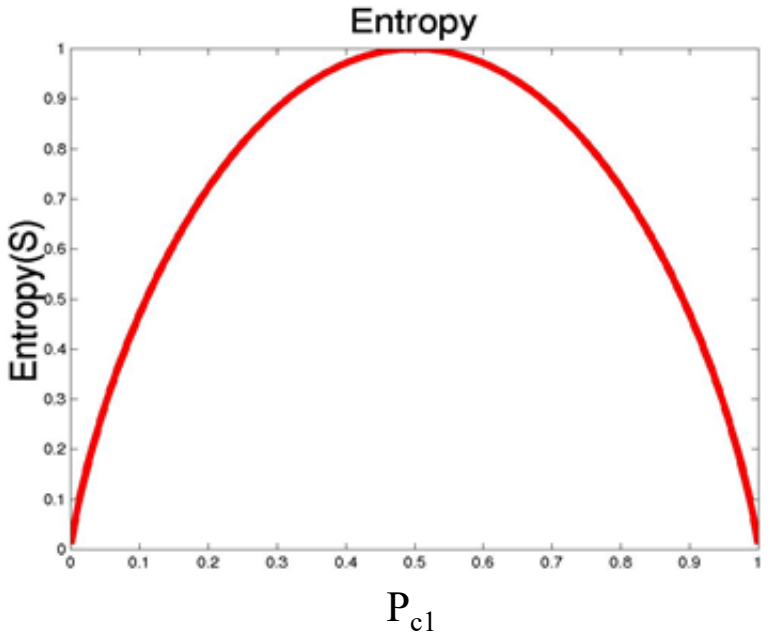
(+, -, +, -, +, -, +, +, +, -, -, +, +, +)

$$\begin{aligned} \text{Entropy(S)} &= -P_{c1}\log_2(P_{c1}) - P_{c2}\log_2(P_{c2}) \\ &= -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) \\ &= 0.940 \end{aligned}$$

Suppose S has all positive or all negative Examples, then Entropy(S) = 0

(+, +, +, +, +, +, +, +, +), or

(-, -, -, -, -, -, -, -, -, -, -, -, -, -)



Entropy is 0 (minimum) if all members belong to the same class. Entropy is 1 (maximum) if the collection consists of equal number of positive and negative examples. Assume: $0\text{Log}_2 0 = 0$.

Calculating entropy

Calculating the previous example:

$$E(S) = -\frac{9}{14} \cdot log_2\left(\frac{9}{14}\right) - \frac{5}{14} \cdot log_2\left(\frac{5}{14}\right)$$

- If your calculator can't work out logs to base 2 then use the following:

$$\log_2(x) = \frac{\log_{10}(x)}{\log_{10}(2)} \approx \frac{\log_{10}(x)}{0.3010}$$

Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
9	5	0.6429	-0.6374	0.3571	-1.4854	0.9403

Information gain

Information gain is the expected reduction in entropy caused by partitioning the examples according to an attribute A.

- Gain(S,A) of an attribute A, relative to a collection of examples S (with v groups having $|S_v|$ elements) is:

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

Entropy before split

Expected entropy after split

How ID3 uses information gain

The algorithm:

- 'Splits' on the attribute that provides the most information gain – that is, gives the purest class breakdown at each step in the decision tree.
- This is the "best" decision attribute in the algorithm following.

Recall: purer class = entropy reduction!

Top-down induction: ID3

The algorithm (*Iterative Dichotomiser 3*):

- At each step, determine the "best" decision attribute, *A*.
- Assign *A* as decision attribute for node.
- For each value of *A* create new descendant.
- Sort training examples according to the attribute value.
- If all training examples are homogenous (i.e., perfectly classified), stop, else iterate over new leaf nodes.

For this algorithm assume we class attribute is categorical.

en.wikipedia.org/

Example: playing tennis

Build a decision tree for playing tennis based on weather conditions.

Training set (S):

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

First split on: Outlook, Temperature, Humidity or Wind?

Playing tennis: initial entropy

Training set (*S*): Initial entropy before splitting based on 9 Yes/5 No:

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
9	5	0.6429	-0.6374	0.3571	-1.4854	0.9403

Initial entropy

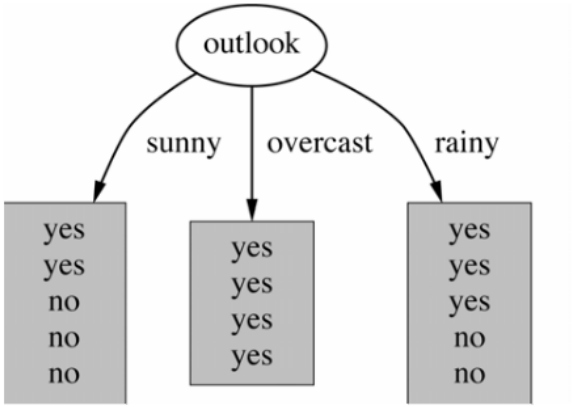
- Without any knowledge of the weather there are 9 Yes and 5 No cases. Initial entropy is:
- $E(S) = -\frac{9}{14} \cdot log_2\left(\frac{9}{14}\right) - \frac{5}{14} \cdot log_2\left(\frac{5}{14}\right)$
- $E(S) = 0.9403$

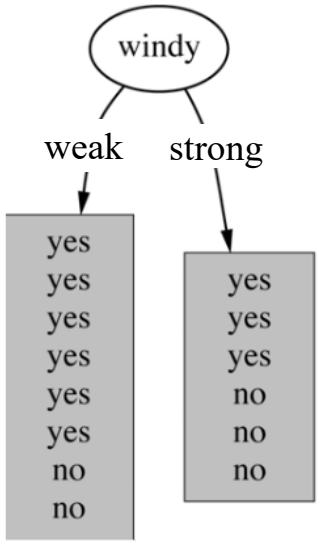
Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
9	5	0.6429	-0.6374	0.3571	-1.4854	0.9403

Which attribute to select?

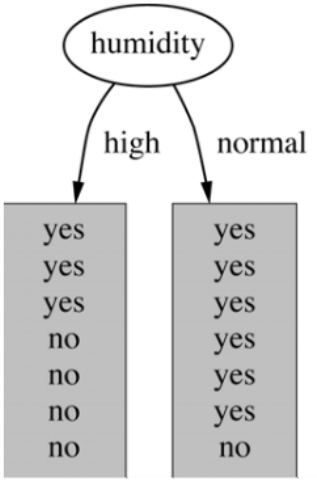
Remember - ID3 chooses the attribute which gives the greatest information gain (reduction in Entropy), or the 'purest' result.

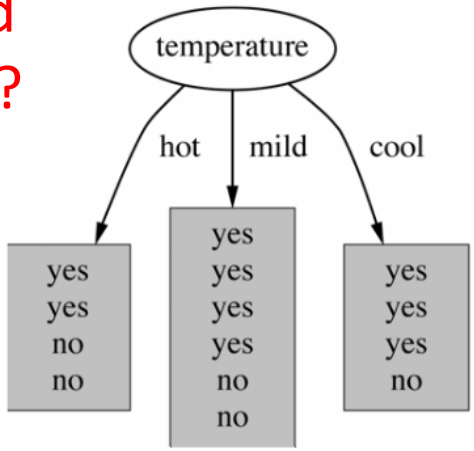
We next calculate the information gain for each attribute in turn.





What would you choose?

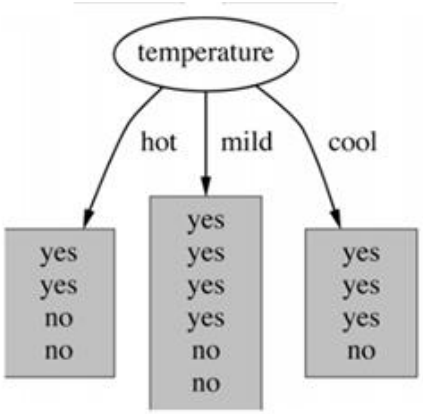




Information gain: Temperature

Calculate entropy for each branch first:

- $E(S_{hot}) = -\frac{2}{4} \cdot log_2\left(\frac{2}{4}\right) - \frac{2}{4} \cdot log_2\left(\frac{2}{4}\right) = 1$
- $E(S_{mild}) = -\frac{4}{6} \cdot log_2\left(\frac{4}{6}\right) - \frac{2}{6} \cdot log_2\left(\frac{2}{6}\right) = 0.918$
- $E(S_{cool}) = -\frac{3}{4} \cdot log_2\left(\frac{3}{4}\right) - \frac{1}{4} \cdot log_2\left(\frac{1}{4}\right) = 0.811$



Now calculate expected entropy and information gain

- $Gain(S, Temp) = E(S) - E(S, Temp)$
- $Gain(S, Temp) = E(S) - \left(\frac{4}{14}1 + \frac{6}{14}0.918 + \frac{4}{14}0.811\right)$
- $= 0.9403 - 0.910$
- $= 0.0292$

Expected entropy is the sum of entropy * probability for each branch.

Information gain: Temperature

As a spreadsheet showing initial entropy and subsequent information gain:

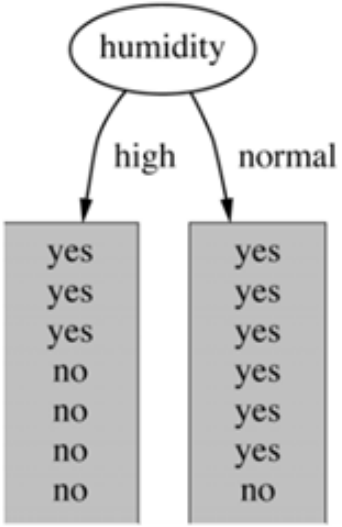
Initial State	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Entropy(S)	9	5	0.6429	-0.6374	0.3571	-1.4854	0.9403

Temperature	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Hot	2	2	0.5000	-1.0000	0.5000	-1.0000	1.0000
Mild	4	2	0.6667	-0.5850	0.3333	-1.5850	0.9183
Cool	3	1	0.7500	-0.4150	0.2500	-2.0000	0.8113
EEntropy(Temp)							0.9111
Gain(S, Temp)							**0.0292**

Information gain: Humidity

Calculate entropy for each branch first:

- $E(S_{high}) = -\frac{3}{7} \cdot log_2\left(\frac{3}{7}\right) - \frac{4}{7} \cdot log_2\left(\frac{4}{7}\right) = 0.9852$
- $E(S_{normal}) = -\frac{6}{7} \cdot log_2\left(\frac{6}{7}\right) - \frac{1}{7} \cdot log_2\left(\frac{1}{7}\right) = 0.5917$



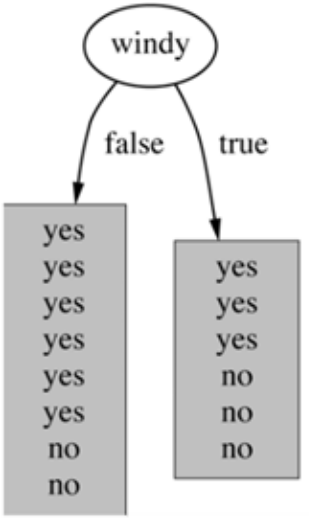
Now calculate expected entropy and information gain

- $Gain(S, Humidity) = E(S) - E(S, Humidity)$
- $Gain(S, Humidity) = E(S) - \left(\frac{7}{14}0.9852 + \frac{7}{14}0.5917\right)$
- $= 0.9403 - 0.7885$
- $= 0.1518$

Information gain: Windy (for you to do)

Calculate entropy for each branch first:

- $E(S_{false}) = -\frac{6}{8} \cdot log_2\left(\frac{6}{8}\right) - \frac{2}{8} \cdot log_2\left(\frac{2}{8}\right) =$ ☐
- $E(S_{true}) = -\frac{3}{6} \cdot log_2\left(\frac{3}{6}\right) - \frac{3}{6} \cdot log_2\left(\frac{3}{6}\right) =$ ☐



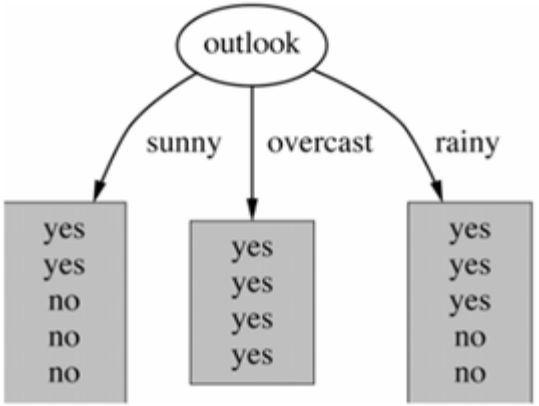
Now calculate expected entropy and information gain

- $Gain(S, Windy) = E(S) - E(S, Windy)$
- $Gain(S, Windy) = E(S) - \left(\frac{8}{14}$ ☐ $+ \frac{6}{14}$ ☐ $\right) =$ ☐
- $= 0.9403 -$ ☐ $=$ ☐

Information gain: Outlook (for you to do)

Calculate entropy for each branch first:

- $E(S_{sunny}) = -\frac{2}{5} \cdot log_2\left(\frac{2}{5}\right) - \frac{3}{5} \cdot log_2\left(\frac{3}{5}\right) =$ ☐
- $E(S_{overcast}) = 0$
- $E(S_{rainy}) = -\frac{3}{5} \cdot log_2\left(\frac{3}{5}\right) - \frac{2}{5} \cdot log_2\left(\frac{2}{5}\right) =$ ☐



Now calculate expected entropy and information gain

- $Gain(S, Outlook) = E(S) - E(S, Outlook)$
- $Gain(S, Outlook) = E(S) - \left(\frac{5}{14}$ ☐ $+ \frac{4}{14}$ ☐ $+ \frac{5}{14}$ ☐ $\right)$
- $= 0.9403 -$ ☐ $=$ ☐

Calcs: Humidity, Windy, Outlook

Humidity	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
High	3	4	0.4286	-1.2224	0.5714	-0.8074	0.9852
Normal	6	1	0.8571	-0.2224	0.1429	-2.8074	0.5917
EEntropy(Humidity)							0.7885
Gain(S,□Humidity)							**0.1518**

Wind	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Weak	6	2	0.7500	-0.4150	0.2500	-2.0000	0.8113
Strong	3	3	0.5000	-1.0000	0.5000	-1.0000	1.0000
EEntropyWind)							0.8922
Gain(S,□Wind)							**0.0481**

Outlook	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Sunny	2	3	0.4000	-1.3219	0.6000	-0.7370	0.9710
Overcast	4	0	1.0000	0.0000	0.0000	0.0000	0.0000
Rain	3	2	0.6000	-0.7370	0.4000	-1.3219	0.9710
EEntropy(Outlook)							0.6935
Gain(S,□Outlook)							**0.2467**

Attribute giving greatest information gain

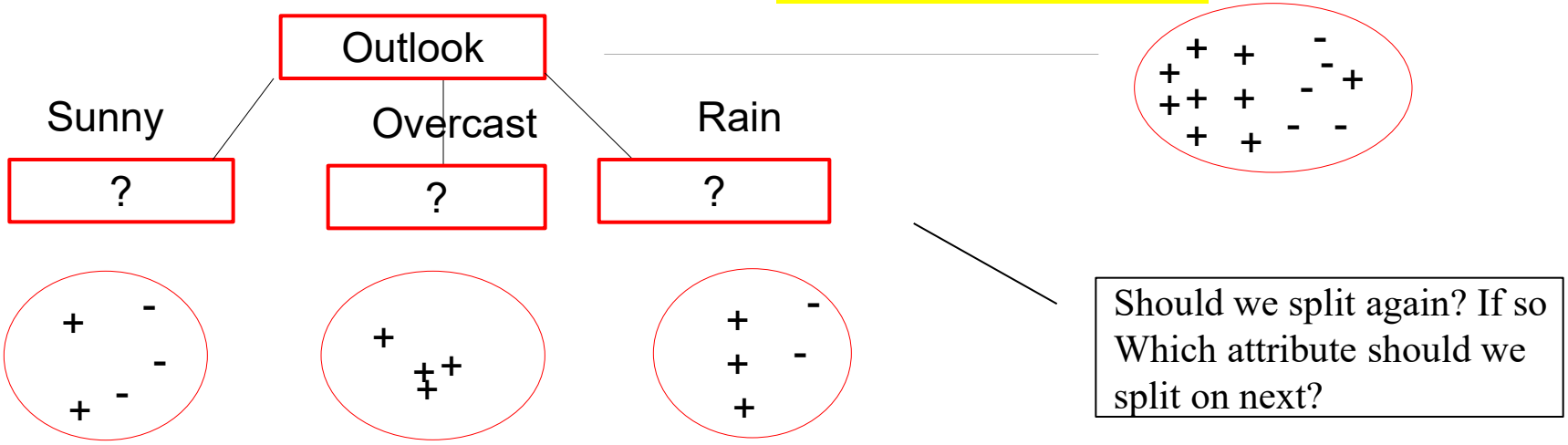
Which attribute to choose? Outlook, Temperature, Humidity or Wind?

Gain(S, Temperature) = 0.029

Gain(S, Humidity) = 0.151

Gain(S, Wind) = 0.048

Gain(S, Outlook) = 0.247



Entropy after "Outlook"

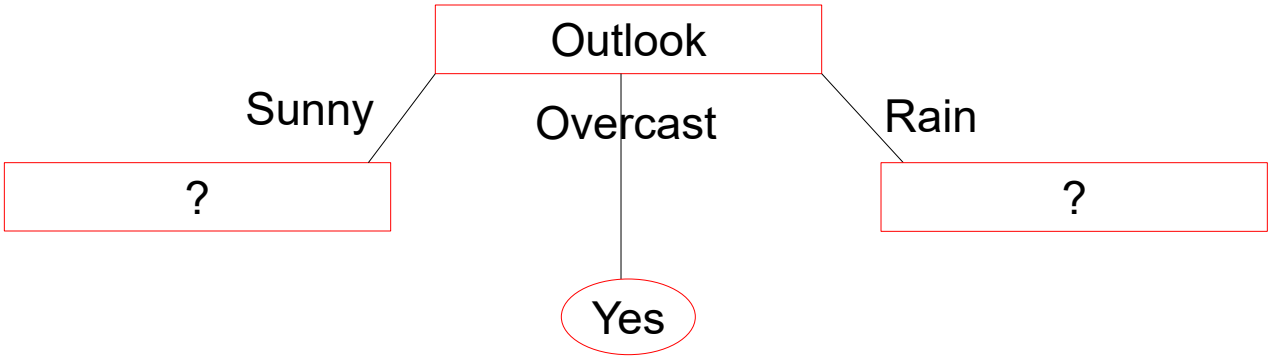
The entropy of each branch of the decision tree after split on Outlook is shown below.

- Information gain in descendent trees is now measured as change in the entropy of each branch.
- For example, Entropy(Sunny) = 0.971

Outlook	Yes	No	P(Yes)	og2(Yes)	P(No)	.og2(No)	Entropy
Sunny	2	3	0.400	-1.322	0.600	-0.737	0.971
Overcast	4	0	1.000	0.000	0.000	0.000	0.000
Rain	3	2	0.600	-0.737	0.400	-1.322	0.971
EEntropy(Outlook)							0.694

Which attribute to split on next?

Now, starting with Sunny, which attribute should be split on next? Temperature, Humidity or Wind?

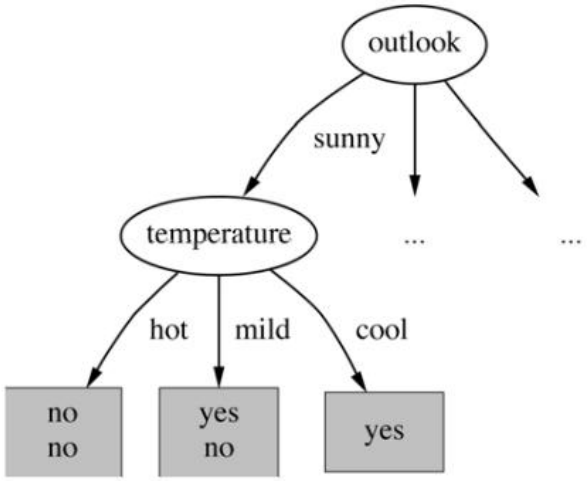


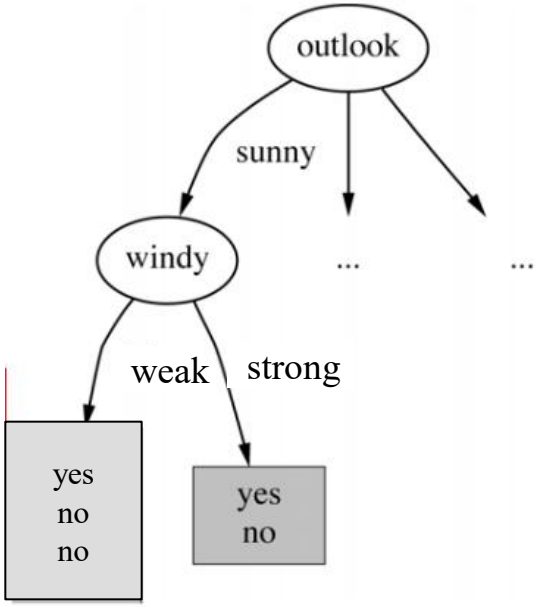
ID3 Step 2: gain(S_sunny, ???)

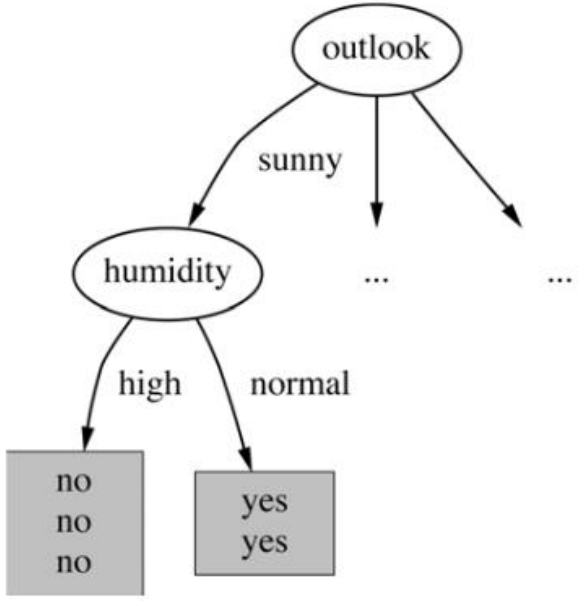
Now consider subset corresponding to “Sunny”:

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Which attribute to split on next?







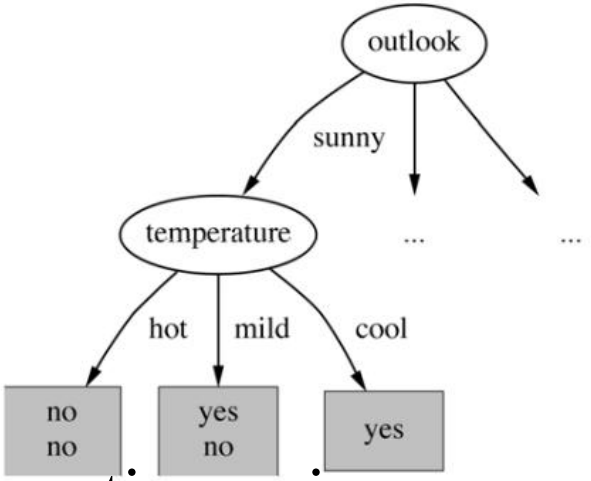
What attribute would you choose?

Do you need to do any calculations?

ID3 Step 2: Gain Sunny, Temperature

Calculate entropy for each branch first:

- $E(S_{sunny,\,hot}) = -\frac{0}{2} \cdot log_2\left(\frac{0}{2}\right) - \frac{2}{2} \cdot log_2\left(\frac{2}{2}\right) = 0$
- $E(S_{sunny,mild}) = -\frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) - \frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) = 1$
- $E(S_{sunny,cool}) = -\frac{1}{1} \cdot log_2\left(\frac{1}{1}\right) - \frac{0}{1} \cdot log_2\left(\frac{0}{1}\right) = 0$



Now calculate expected entropy and information gain

- $Gain(S, Sunny, Temp) = E(S, Sunny) - E(S, Sunny, Temp)$
- $Gain(S, Sunny, Temp) = E(S, Sunny) - \left(\frac{2}{5}0 + \frac{2}{5}1 + \frac{1}{5}0\right)$
- $= 0.971 - 0.4 = 0.571$

Calculations for all attributes shown on the next slide…

ID3 Step 2: Gain for Sunny Outlook

Sunny, Temp	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Hot	0	2	0.0000	0.0000	1.0000	0.0000	0.0000
Mild	1	1	0.5000	-1.0000	0.5000	-1.0000	1.0000
Cool	1	0	1.0000	0.0000	0.0000	0.0000	0.0000
EEntropy(Temp)							0.4000
Gain(Sunny, Temp)							**0.5710**

Sunny, Humid	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
High	0	3	0.0000	0.0000	1.0000	0.0000	0.0000
Normal	2	0	1.0000	0.0000	0.0000	0.0000	0.0000
EEntropy(Temp)							0.0000
Gain(Sunny, Humid)							**0.9710**

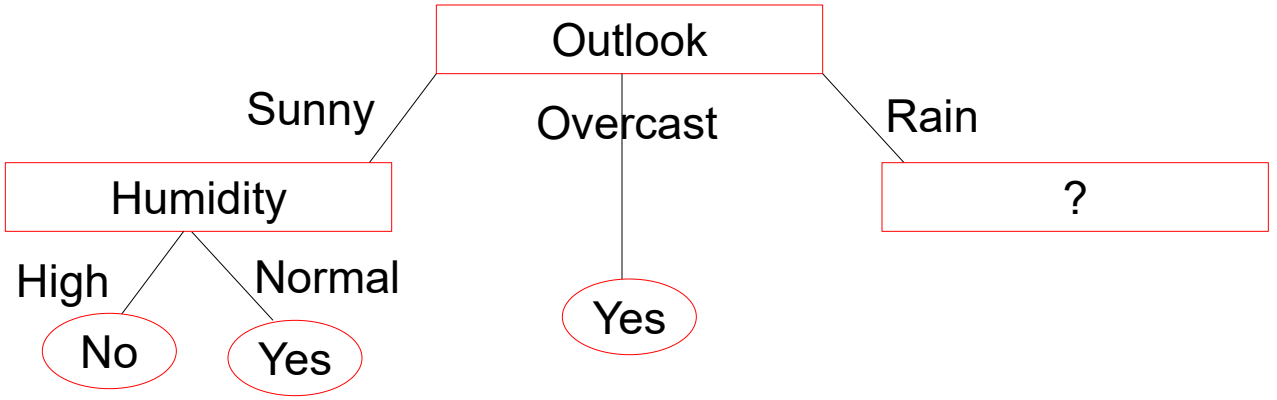
Sunny, Wind	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Weak	1	2	0.3333	-1.5850	0.6667	-0.5850	0.9183
Strong	1	1	0.5000	-1.0000	0.5000	-1.0000	1.0000
EEntropyWind)							0.9510
Gain(Sunny, Wind)							**0.0200**

ID3 Step 2: Gain for Sunny Outlook

Which attribute to choose? Temperature, Humidity or Wind?

- $Gain(S_{sunny}, Wind) = 0.020$
- $Gain(S_{sunny}, Humidity) = 0.971$
- $Gain(S_{sunny}, Temperature) = 0.570$

Choose this one!



Repeat the process to find which attribute is the best to split on at this step

Not all leaves need to be 'pure'.
Splitting stops when the data can't be split any further.

For you to do

Now consider subset after “Rain Outlook”:

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

For you to do

Now consider subset after “Rain Outlook”:

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

For you to do

Class counts and expected entropy after rain outlook.

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Rain, Temp	Yes	No
Hot		
Mild		
Cool		

Rain, Humid	Yes	No
High		
Normal		

Rain, Wind	Yes	No
Weak		
Strong		

For you to do

Class counts and expected entropy after rain outlook.

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Rain,Temp	Yes	No
Hot		
Mild		
Cool		

Rain,Humid	Yes	No
High		
Normal		

Rain,Wind	Yes	No
Weak	3	0
Strong	0	2

What would you choose?

ID3 Step 3: Gain for Rain Outlook

Rain, Temp	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Hot							
Mild							
Cool							
EEntropy(Temp)							
Gain(Rain, Temp)							

$$\text{Entropy}(S) = -P_{C_1}\log_2\left(P_{C_1}\right) - P_{C_2}\log_2\left(P_{C_2}\right) = -\frac{N_{C_1}}{N}\log_2\frac{N_{C_1}}{N} - \frac{N_{C_2}}{N}\log_2\frac{N_{C_2}}{N}$$

Rain, Humid	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
High							
Normal							
EEntropy(Temp)							
Gain(Rain, Humid)							

$$\log_2(x) = \frac{\log_{10}(x)}{\log_{10}(2)} \approx \frac{\log_{10}(x)}{0.3010}$$

Rain, Wind	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Weak							
Strong							
EEntropyWind)							
Gain(Rain, Wind)							

ID3 Step 3: Gain for Rain Outlook

Rain, Temp	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Hot	0	0	0.0000	0.0000	0.0000	0.0000	0.0000
Mild	2	1	0.6667	-0.5850	0.3333	-1.5850	0.9183
Cool	1	1	0.5000	-1.0000	0.5000	-1.0000	1.0000
EEntropy(Temp)							0.9510
Gain(Rain, Temp)							**0.0200**

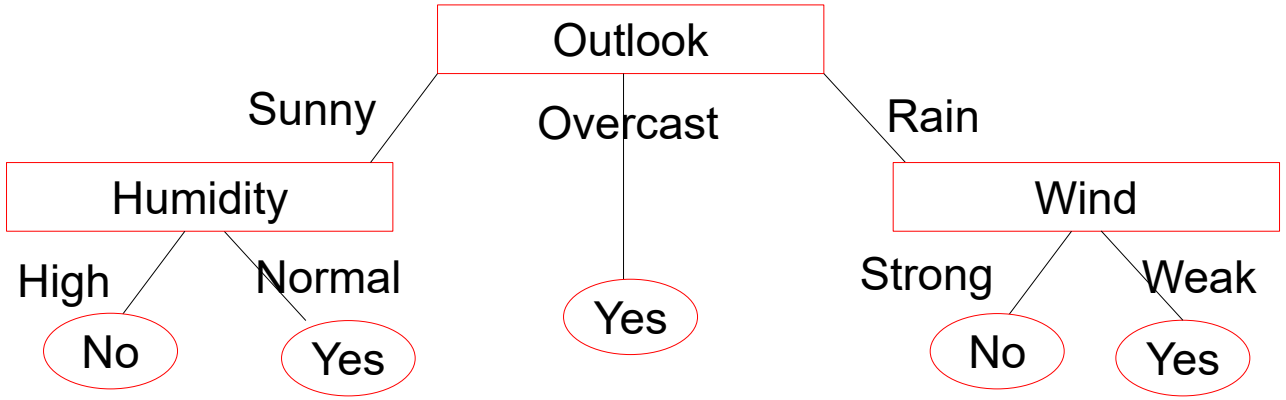
Rain, Humid	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
High	1	1	0.5000	-1.0000	0.5000	-1.0000	1.0000
Normal	2	1	0.6667	-0.5850	0.3333	-1.5850	0.9183
EEntropy(Temp)							0.9510
Gain(Rain, Humid)							**0.0200**

Rain, Wind	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Weak	3	0	1.0000	0.0000	0.0000	0.0000	0.0000
Strong	0	2	0.0000	0.0000	1.0000	0.0000	0.0000
EEntropyWind)							0.0000
Gain(Rain, Wind)							**0.9710**

The Final tree

The process of selecting a new attribute and partitioning the training examples is repeated for each non-leaf node, using only the examples associated with that node. Attributes that have been incorporated higher in the tree are excluded, so that any given attribute can appear at most once along any path through the tree.
The process continues for each leaf node until:

- every attribute has been included along that path through the tree or
- the training examples associated with this leaf node all have the same class.



The Decision Tree Rules

In addition to generating a tree structure, explicit rules for classifying 'play/don't play' are also generated:

If outlook = Overcast Then Play= Yes {No=0, Yes=4}

If outlook = Rain And wind = Strong Then Play= No {No=2, Yes=0}

If outlook = Rain And wind = Weak Then Play = Yes {No=0, Yes=3}

If outlook = Sunny And humidity = High Then Play = No {No=3, Yes=0}

If outlook = Sunny And humidity = Normal Then Play = Yes {No=0, Yes=2}

Metrics for Performance Evaluation

How to evaluate the performance of a model?

- Training and testing
- Confusion matrix
- Cross validation

Training and testing

- You can measure a classifier's performance in terms of the error rate (proportion of errors made over a set of instances).
- However, low error on the training data is not a good measure. To predict the performance of a classifier on a new data, we need to assess its error on data that was not used to build the model.
- In general, the data set is divided into two subsets: training and testing. Training for learning the model and testing for determining how well it will do on unseen data.

Performance evaluation of the Model

Focus on the predictive capability of a model

- Rather than how fast it takes to classify or build models, scalability, etc.

How to determine accuracy of decision tree in classifying/predicting?

- Usual to have two data sets:
  - A *Training Set* and a *Test Set*
  - This can be created by dividing the data set into two sets – e.g. 70%/30%
- We create the decision tree model using the training set.
- Then run the test set through the model to find out what the predicted class is.
- Then compare the predicted class with the actual class to see how accurate the model is.

Metrics for Performance Evaluation

One way of assessing performance is to calculate accuracy based on a *confusion matrix* (for the test data classification).

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive) b: FN (false negative)

c: FP (false positive) d: TN (true negative)

Metrics for Performance Evaluation

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

Most widely-used metric:

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

Also:

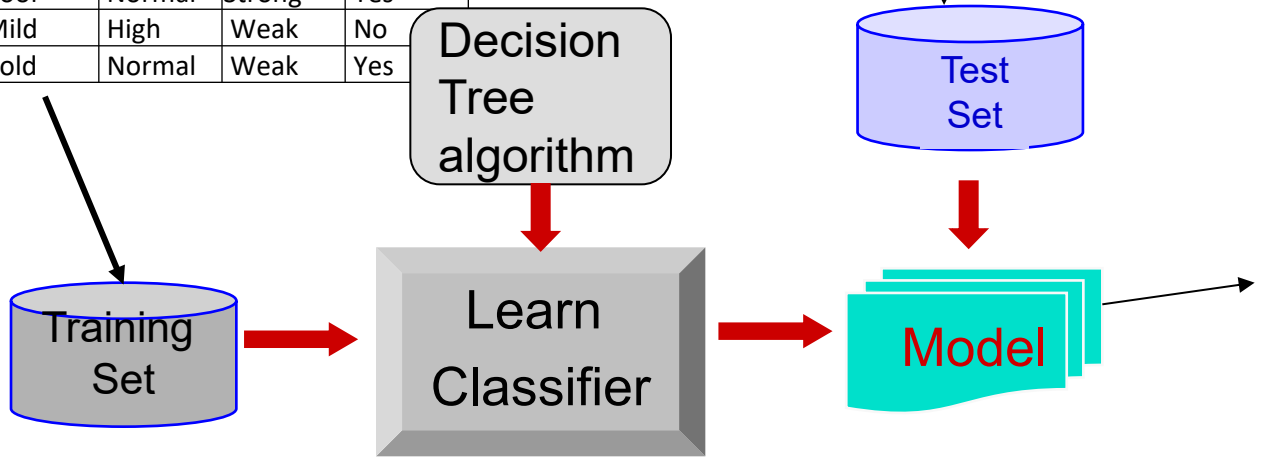
Precision = TP/(TP + FP)

Sensitivity = TP /(TP + FN)

Play Tennis example

ID	outlook	temp	humidity	wind	play
D1	Sunny	Hot	High	Weak	No
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes

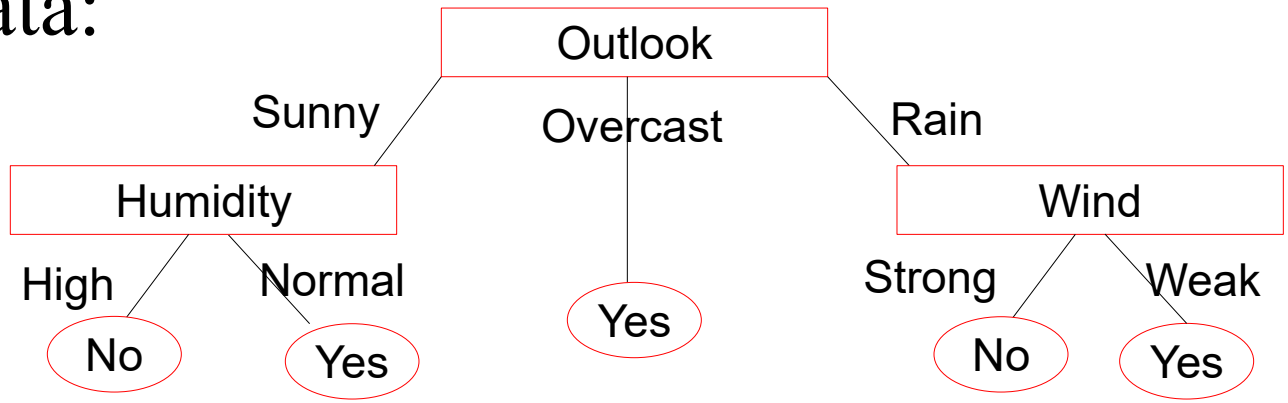
ID	outlook	temp	humidity	wind	play
D15	Sunny	Mild	Normal	Strong	Yes
D16	Sunny	Hot	High	Weak	No
D17	Rain	Hot	High	Weak	Yes
D18	Overcast	Cool	High	Strong	Yes
D19	Overcast	Mild	Normal	Weak	Yes
D20	Rain	Mild	Normal	Weak	Yes



		Predicted class	
		Class=Yes	Class=No
actual class	Class=Yes	a	b
	Class=No	c	d

Play Tennis example

Let's see what our model would predict using the test data:



Day	Outlook	Temperature	Humidity	Wind	Play	Predict
D15	Sunny	Mild	Normal	Strong	No	yes
D16	Sunny	Hot	High	Weak	Yes	no
D17	Rain	Hot	High	Weak	No	yes
D18	Overcast	Cool	High	Strong	No	yes
D19	Overcast	Mild	Normal	Weak	Yes	yes
D20	Rain	Mild	Normal	Weak	Yes	yes

Metrics for Performance Evaluation…

		PREDICTED CLASS	
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	2 (TP)	1 (FN)
	Class=No	3 (FP)	0 (TN)

The most widely-used metric:

$$Accuracy = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN} = \frac{2+0}{6} = 33.3\%$$

More ways to measure classification performance next lecture...

Fitting a decision tree in R

In this section we'll cover:

- Fitting a decision tree in R
- Evaluating tree performance

Decision trees in R

There are a number of packages to create decision trees in R. We will start with the "tree" package.

```
> install.packages("tree")
> library(tree)
```

- *Note: "tree" aims to minimise 'impurity' by binary splitting. Similar, but not identical, to ID3 in function.*

https://cran.r-project.org/web/packages/tree/index.html

Classification tree: data

Build and test a model using the playtennis data.

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Classification tree: data

Build and test a model using the playtennis data.

- Ensure inputs are factors – not character vars.

  > ptt <- read.csv("playtennistrain.csv", stringsAsFactors = T)

- As the training set has too few instances for the tree package to fit a model, a larger synthetic data set is created by resampling with replacement.

  > set.seed(9999) #make random selection repeatable

  > # resampling with replacement

  > pttrain = ptt[sample(nrow(ptt), 100, replace = TRUE),]

  Training data has been resampled to make 100 rows.

Classification tree: building the tree

Build the tree using "Play" as the response and all input variables except "Day" as predictors.

Syntax is very similar to linear model function.

Output is a list.

```
> ptfit = tree(Play ~. -Day, data = pttrain)
```

Fit model to all attributes except "Day".

? tree

- Description

  A tree is grown by binary recursive partitioning using the response in the specified formula and choosing splits from the terms of the right-hand-side.

- Usage

```
tree(formula, data, weights, subset,
na.action = na.pass, control =
tree.control(nobs, ...),
method = "recursive.partition",
split = c("deviance", "gini"),
model = FALSE, x = FALSE, y = TRUE, wts = TRUE,
...)
```

tree: details

- **A tree is grown by binary recursive partitioning using the response in the specified formula and choosing splits from the terms of the right-hand-side. Numeric variables are divided into X < a and X > a;**
- **The levels of an unordered factor are divided into two non-empty groups.**
- **The split which maximizes the reduction in impurity is chosen, the data set split, and the process repeated.**
- **Splitting continues until the terminal nodes are too small or too few to be split.**

https://cran.r-project.org/web/packages/tree/index.html

Classification tree: summary

Use “summary” to get a basic idea of model performance: terminal nodes, error measures.

```
> summary(ptfit)
Classification tree:
tree(formula = Play ~ . - Day, data = pttrain)
Number of terminal nodes:  7
Residual mean deviance:  0 = 0 / 93
Misclassification error rate: 0 = 0 / 100
```

Classification tree: details

Details of each split, root to leaf, left to right.

- ptfit

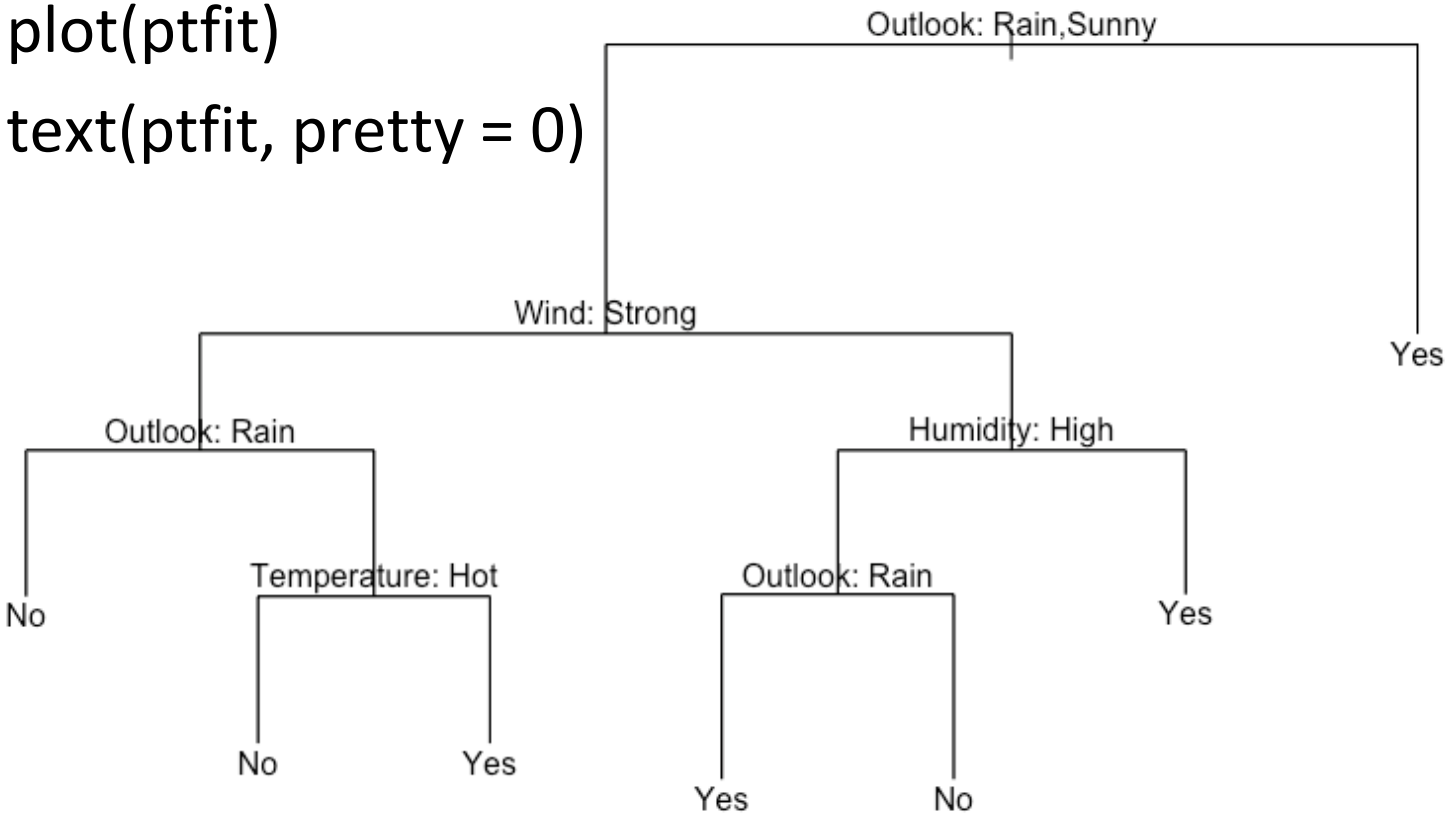
```
> ptfit
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 100 100 Yes ( 0.4 0.6 )
   2) Outlook: Rain,Sunny 70 100 No ( 0.6 0.4 )
     4) Wind: Strong 36  40 No ( 0.8 0.2 )
       8) Outlook: Rain 22   0 No ( 1.0 0.0 ) *
       9) Outlook: Sunny 14  20 Yes ( 0.4 0.6 )
        18) Temperature: Hot 6   0 No ( 1.0 0.0 ) *
        19) Temperature: Mild 8   0 Yes ( 0.0 1.0 ) *
     5) Wind: Weak 34  40 Yes ( 0.3 0.7 )
      10) Humidity: High 18  20 No ( 0.6 0.4 )
        20) Outlook: Rain 7   0 Yes ( 0.0 1.0 ) *
        21) Outlook: Sunny 11   0 No ( 1.0 0.0 ) *
      11) Humidity: Normal 16   0 Yes ( 0.0 1.0 ) *
   3) Outlook: Overcast 30   0 Yes ( 0.0 1.0 ) *
```

Classification tree: plot

Headers give rule for left branching.

```
> plot(ptfit)
> text(ptfit, pretty = 0)
```



Classification tree: testing the model

To test the model, make a prediction for each input from the test data set and cross tabulate with the actual classification in the test data:

```
> pttest <- read.csv("playtennistest.csv",
  stringsAsFactors = T)
> tpredict = predict(ptfit, pttest, type = "class")
> Tpredict
[1] Yes No  Yes Yes Yes Yes
Levels: No Yes
```

Classification tree: testing the model

Comparing predicted with actual values as a Confusion Matrix:

```
> tpredict
[1] Yes No  Yes Yes Yes Yes
> pttest$Play
[1] No  Yes No  No  Yes Yes
> table(observed = pttest$Play, predicted = tpredict)
          predicted
observed No Yes
     No   0   3
     Yes  1   2
```

Edgar Anderson's Iris data

50 samples from 3 species:

- Iris setosa, – virginica, – versicolor

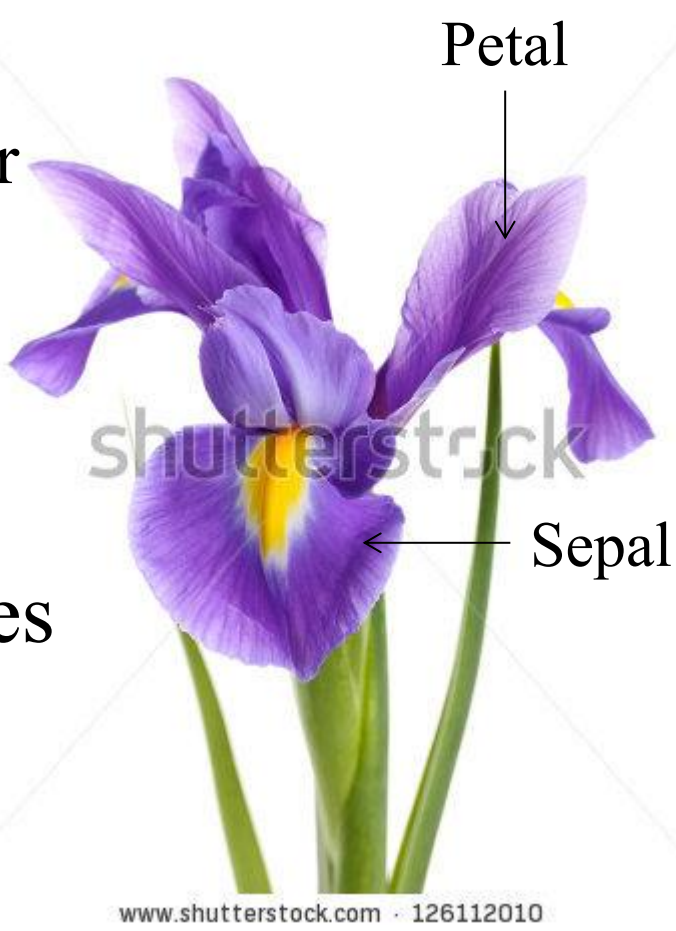
Four features measured:

- Sepal width and length
- Petal width and length

Is it possible to distinguish species using physical measurements?

- Data is packaged with R: "iris"

http://en.wikipedia.org/wiki/Iris_flower_data_set



Regression tree: data

Build and test a model using the iris data.

Subset the data into training and test data sets.

```
> # to select 70% of rows to train, 30% for testing
> set.seed(9999) # make random selection repeatable
> # create vector of row indices for training
> train.row = sample(1:nrow(iris), 0.7*nrow(iris))
> # assign train/test using row index
> iris.train = iris[train.row,]
> iris.test = iris[-train.row,]
```

Row indices for training

Not row indices for training

Regression tree: building and testing

Adapting the same commands used for the tennis example:

```
> itree = tree(Species ~., data = iris.train)
> itree
> summary(itree)
> plot(itree)
> text(itree, pretty = 0)
> ipredict = predict(itree, iris.test, type = "class")
> table(observed = iris.test$Species, predicted = ipredict)
```

Regression tree: summary

Summary of terminal nodes, variables actually used, error measures.

```
> summary(itree)
Classification tree:
tree(formula = Species ~ ., data = iris.train)
Variables actually used in tree construction:
[1] "Petal.Length" "Petal.Width"
Number of terminal nodes:  4
Residual mean deviance:  0.128 = 12.9 / 101
Misclassification error rate: 0.0286 = 3 / 105
```

Regression tree: details

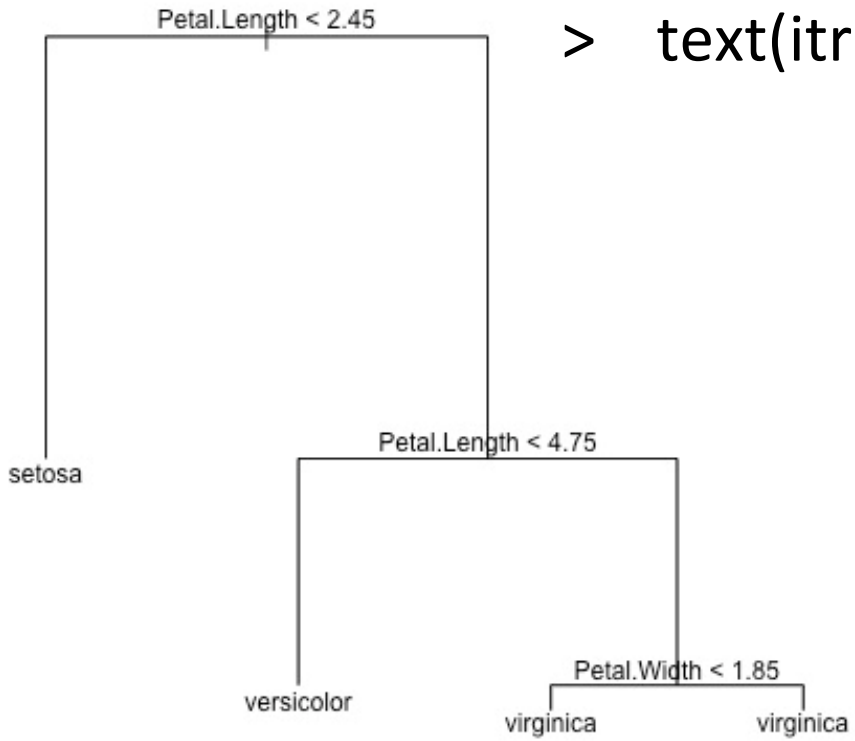
> itree

```
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 105 200 setosa ( 0.36 0.30 0.34 )
   2) Petal.Length < 2.45 38   0 setosa ( 1.00 0.00 0.00 ) *
   3) Petal.Length > 2.45 67  90 virginica ( 0.00 0.46 0.54 )
     6) Petal.Length < 4.75 27   0 versicolor ( 0.00 1.00 0.00 ) *
     7) Petal.Length > 4.75 40  30 virginica ( 0.00 0.10 0.90 )
      14) Petal.Width < 1.7 6   8 virginica ( 0.00 0.50 0.50 ) *
      15) Petal.Width > 1.7 34   9 virginica ( 0.00 0.03 0.97 )
        30) Petal.Length < 4.95 5   5 virginica ( 0.00 0.20 0.80 ) *
        31) Petal.Length > 4.95 29   0 virginica ( 0.00 0.00 1.00 ) *
```

Regression tree: plot

```
> plot(itree)
> text(itree, pretty = 0)
```



Classification tree: testing the model

To test the model, make a prediction for each and draw the confusion matrix.

```
> table(observed = iris.test$Species, predicted = ipredict)
                predicted
observed         setosa versicolor virginica
  setosa             13          0         0
  versicolor          0         14         3
  virginica           0          1        14
```

Consideration

How good is each model?

- Could they be be improved?
- Are they too specific, based on the training set?
- *Note that previous examples are sensitive to the value of the random seed. If this changed the decision tree model and/or accuracy may change.*
- More on decision devlopment and testing as well as other classifcation methods next lecture.

Reading/Notes on the presentation

Further Reading:

- An Introduction to Statistical Learning with applications in R, 2nd Ed, 2021. (Springer Texts in Statistics), James, Witten, Hastie and Tibshirani, Chapter 8 (available on-line from Monash Library)

Notes:

- This presentation contains some material created to accompany: *Introduction to Data Mining*, Tan, Steinbach, Kumar. Pearson Education Inc., 2006.
- Presentation originally created by Dr. Sue Bedingfield.