ROBERT GORDON UNIVERSITY ABERDEEN

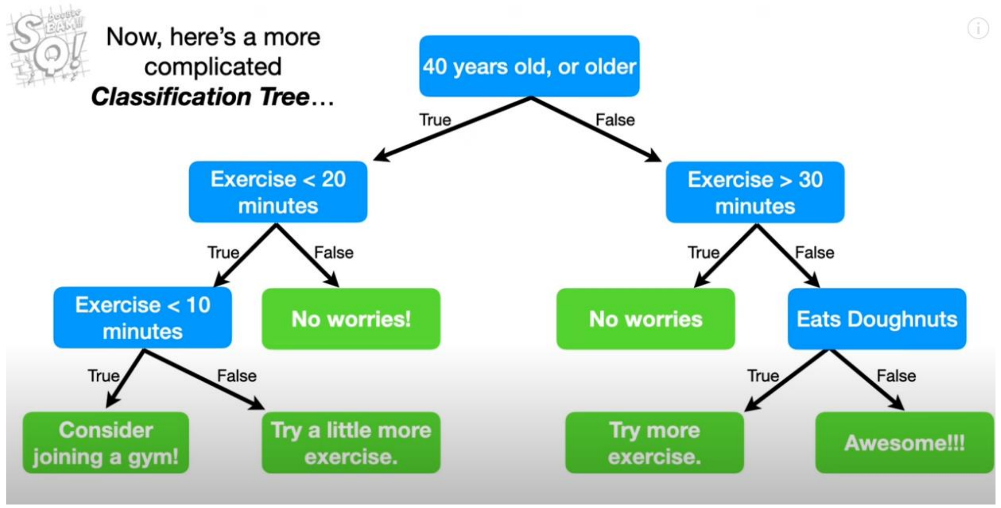
Algorithms: Decision Trees

Also called Classification Trees

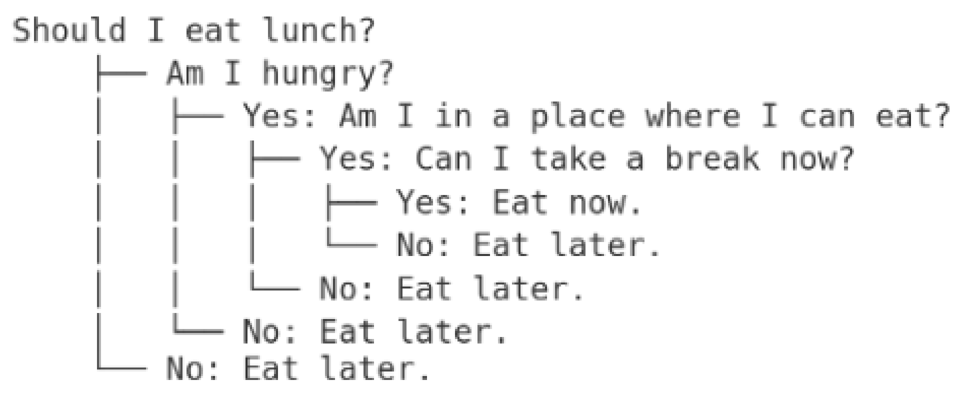
Aside

For an alternative point of view, try StatQuest.

[Decision and Classification Trees, Clearly Explained!!! (youtube.com)]



What is a Decision Tree?



```
Should I eat lunch?
    ├── Am I hungry?
    │   ├── Yes: Am I in a place where I can eat?
    │   │   ├── Yes: Can I take a break now?
    │   │   │   ├── Yes: Eat now.
    │   │   │   └── No: Eat later.
    │   │   └── No: Eat later.
    │   └── No: Eat later.
    └── No: Eat later.
```

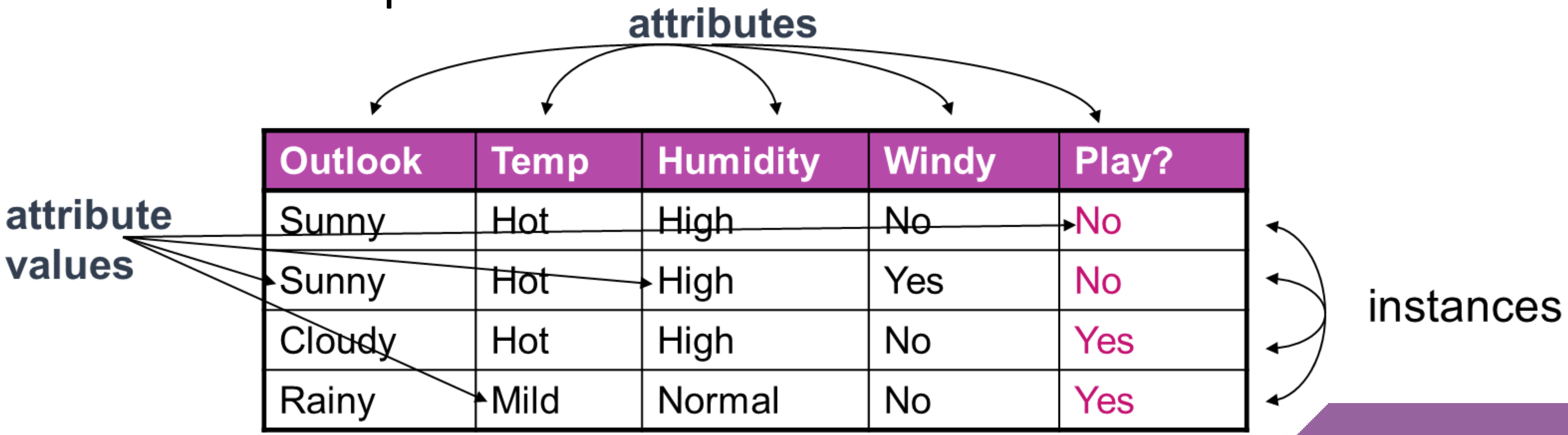
- Method of classification (or regression)
- How to turn a table into “if, then, else” statements
- Decent classifier
- “Explainable” (through counterfactuals)

Outline

- Pre-processing
- Divide & Conquer Algorithms
- ID3 (original algorithm) , C4.5 (improved algorithm) and C5.0 (commercial version when it came out) use Information Gain
  - Choosing root attribute
  - Choosing attributes for subtrees
- Other methods
  - Gini index - CART
  - Chi-squared - CHAID
- Summary

Reminder

- Instance: a single example of a concept (what we are trying to learn)
  - described by a set of attributes (features, columns)
- Attribute: describes a specific characteristic of an instance e.g. age, salary,
- Attribute value: a specific value for an instance attribute



Outlook	Temp	Humidity	Windy	Play?
Sunny	Hot	High	No	No
Sunny	Hot	High	Yes	No
Cloudy	Hot	High	No	Yes
Rainy	Mild	Normal	No	Yes

Pre-processing

Before algorithms are used, there is often a certain amount of pre-processing of the data

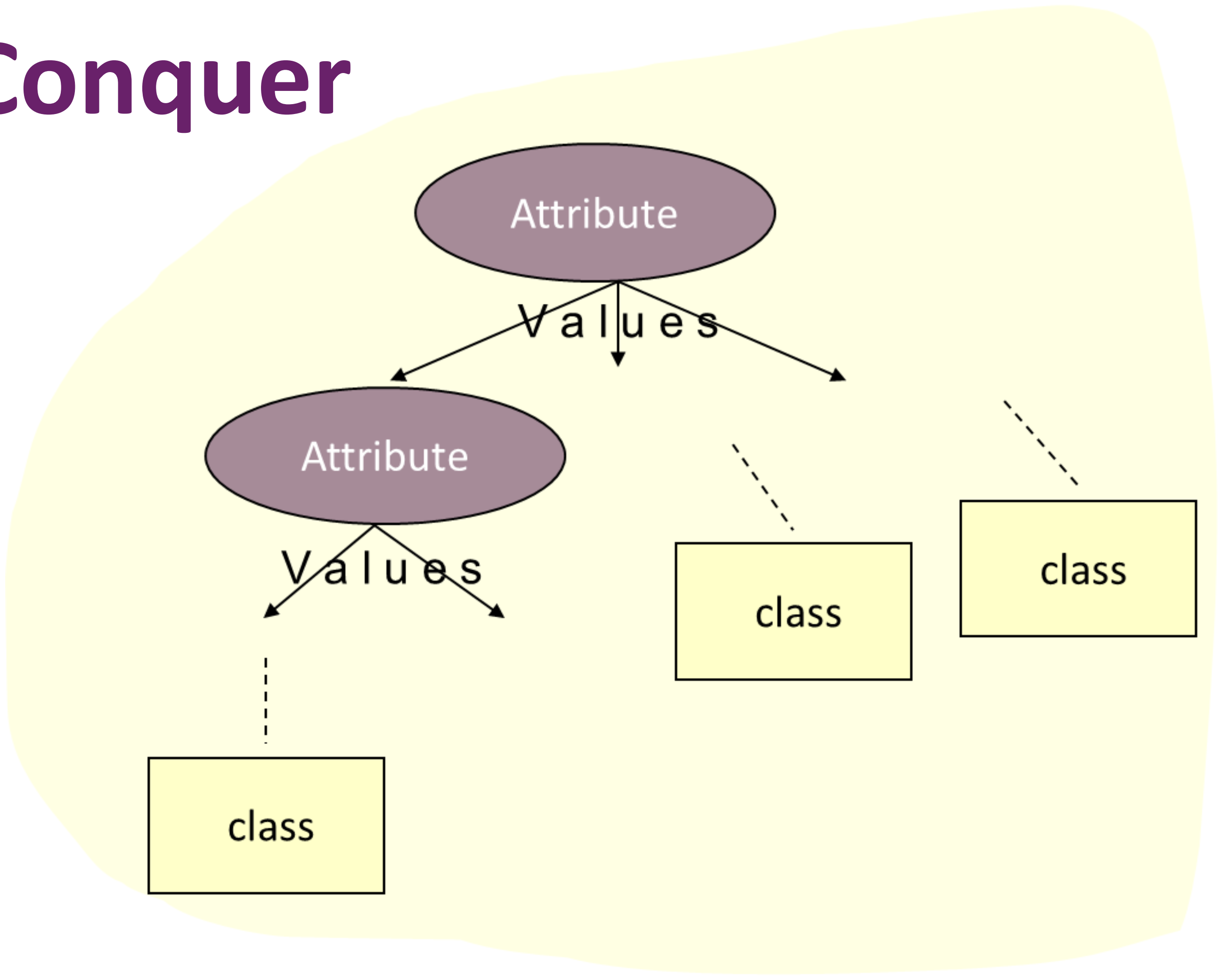
Types of pre-processing

- **Dealing with missing values:**
  - **Value imploding:** replace missing values, e.g. by the mean or median value, by the "nearest neighbour's" value.
  - **Instance removal:** delete whole instance when it contains missing values.
- **Attribute selection**: decide which attributes the algorithm should use.
- …

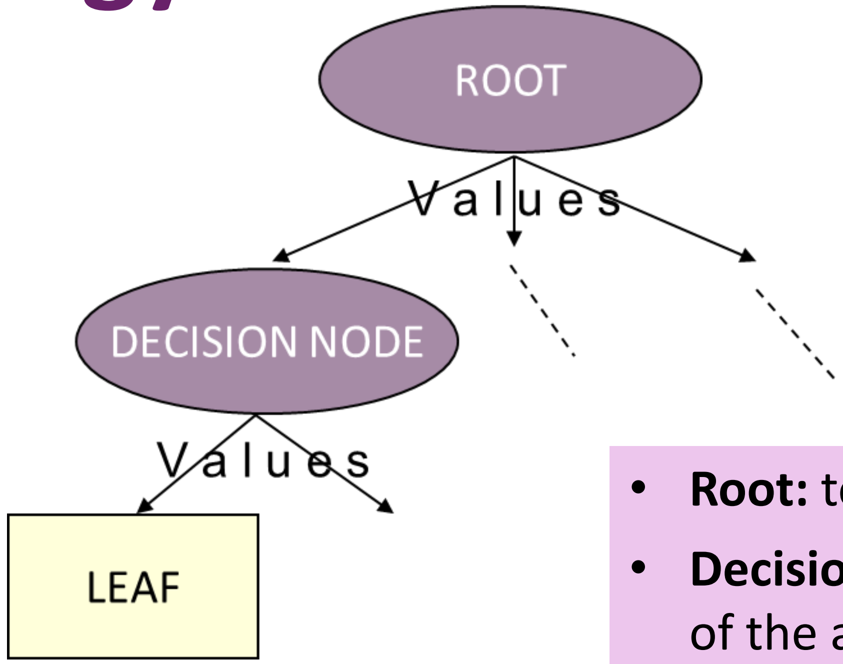
… pre-processing (continued)

- **One-hot encoding**: transform nominal attributes into a set of dummy binary (Boolean) attributes.
- **Dealing with different scales – numeric values:**
  - **Normalisation**: transform numeric attribute values to a scale [0,1]
    - Used when algorithm does not make assumptions about data distribution
  - **Center**: transform the data to have a mean of zero by subtracting the mean from all values.
  - **Scale**: divide each value by the standard deviation for the attribute data.
  - **Standardise:** center + scale– transform numeric values so that the mean is zero and the standard deviation is 1.
    - Used when algorithm assumes data has a normal distribution (bell shaped).
- YOU WILL SEE THESE IN CMM535.

Divide & Conquer



Terminology



- **Root:** top node. Contains all instances
- **Decision node:** divides instances according to the values of the attribute they represent
- **Leaf node:** does not divide instances. Instead it states class for instances in node.

ROBERT GORDON UNIVERSITY ABERDEEN

Pros and cons

- **Advantages**
  - Comprehensible models – easy to understand.
  - Facilitates data exploration.
    - Important attribute identification.
      - If attribute is important it is likely to appear in the tree.
      - The higher up it appears (closer to root) the more important the attribute.
  - Can cope with noise (incorrect values) and outliers to a degree – but see disadvantages.
- **Disadvantages**
  - **Overfitting:** trying too hard to fit the data, leading to more complex, less accurate trees.

Building Decision Trees

- Top-Down Induction of Decision Trees TDIDT most extensively studied machine learning method for data mining.
- **Divide and Conquer** strategy
  - Choose attribute for root node and branch for each value of that attribute.
  - Split instances according to branches.
  - Repeat process for each branch until all instances in the branch have the same class.
- **Assumption:** simplest tree which classifies the instances is best.

Splitting – which attribute first (next)?

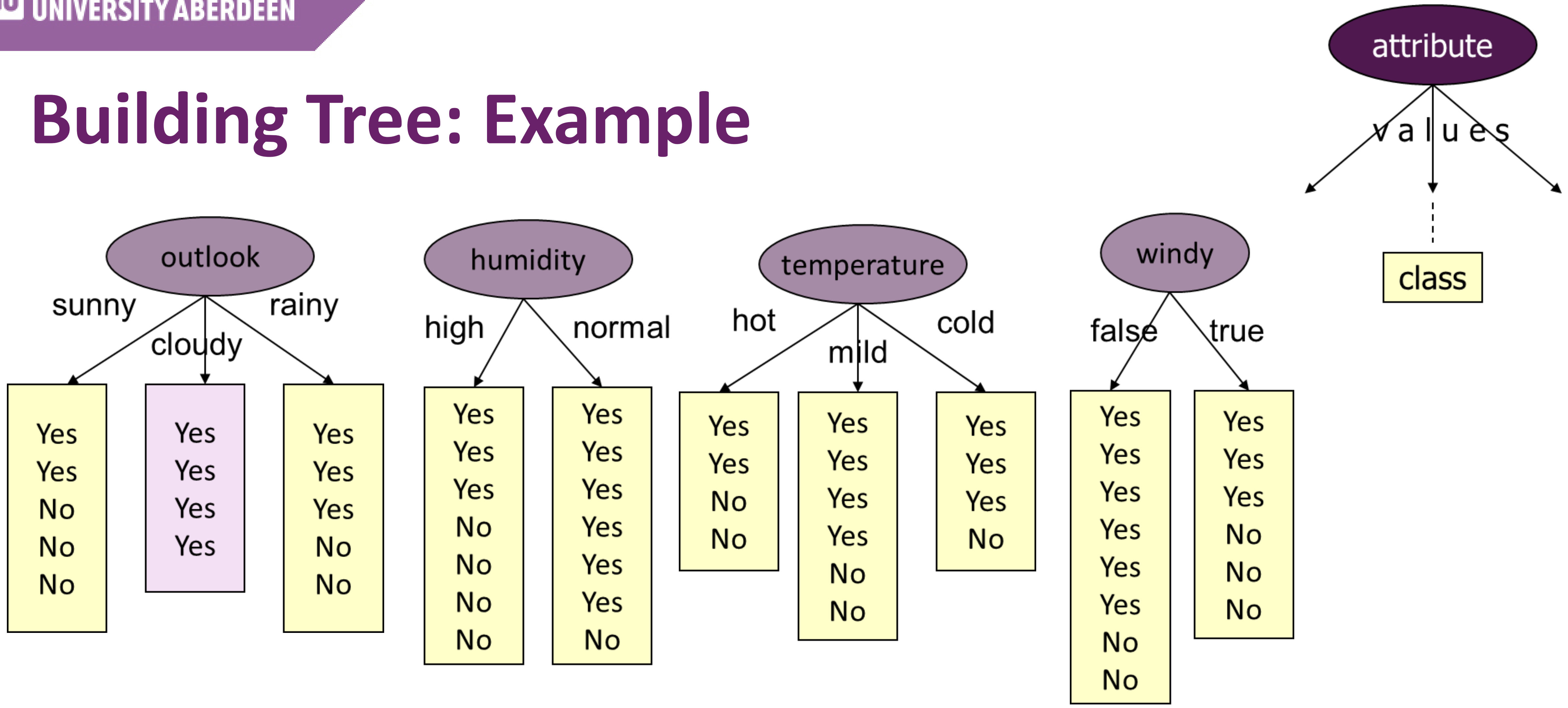
- Various methods
    - **Gini** index: measures difference in diversity/purity of node before and after splitting.
        - Lower values are best – indicate purity improvement from split.
        - **CART** algorithm.
    - **Chi-square** – choose high values (= low correlation)
        - Needs binary class
        - **CHAID** algorithm
    - **Information gain** – measures purity of node(s) before/after splitting
        - Uses entropy (node purity) – low values indicate higher purity after split.
        - **C5.0** algorithm.
        - Choose highest gain – i.e. lowest entropy.

ROBERT GORDON UNIVERSITY ABERDEEN

Example: weather data

Instance	Outlook	Temperature	Humidity	Windy	Play
1	Sunny	Hot	High	False	No
2	Sunny	Hot	High	True	No
3	Cloudy	Hot	High	False	Yes
4	Rainy	Mild	High	False	Yes
5	Rainy	Cool	Normal	False	Yes
6	Rainy	Cool	Normal	True	No
7	Cloudy	Cool	Normal	True	Yes
8	Sunny	Mild	High	False	No
9	Sunny	Cool	Normal	False	Yes
10	Rainy	Mild	Normal	False	Yes
11	Sunny	Mild	Normal	True	Yes
12	Cloudy	Mild	High	True	Yes
13	Cloudy	Hot	Normal	False	Yes
14	Rainy	Mild	High	True	No

Building Tree: Example



Which attribute is best for the root of the tree?

Selecting an attribute with ID3, C4.5 or C5.0

- Which is the best attribute?
  - Select attribute which results in the smallest tree
  - Heuristic
    - select attribute that produces the “purest” nodes
- Purity criterion: information gain
  - Information gain increases with the average purity of the subsets that an attribute produces
- C4.5 and 5.0 choose the attribute which gives greatest information gain

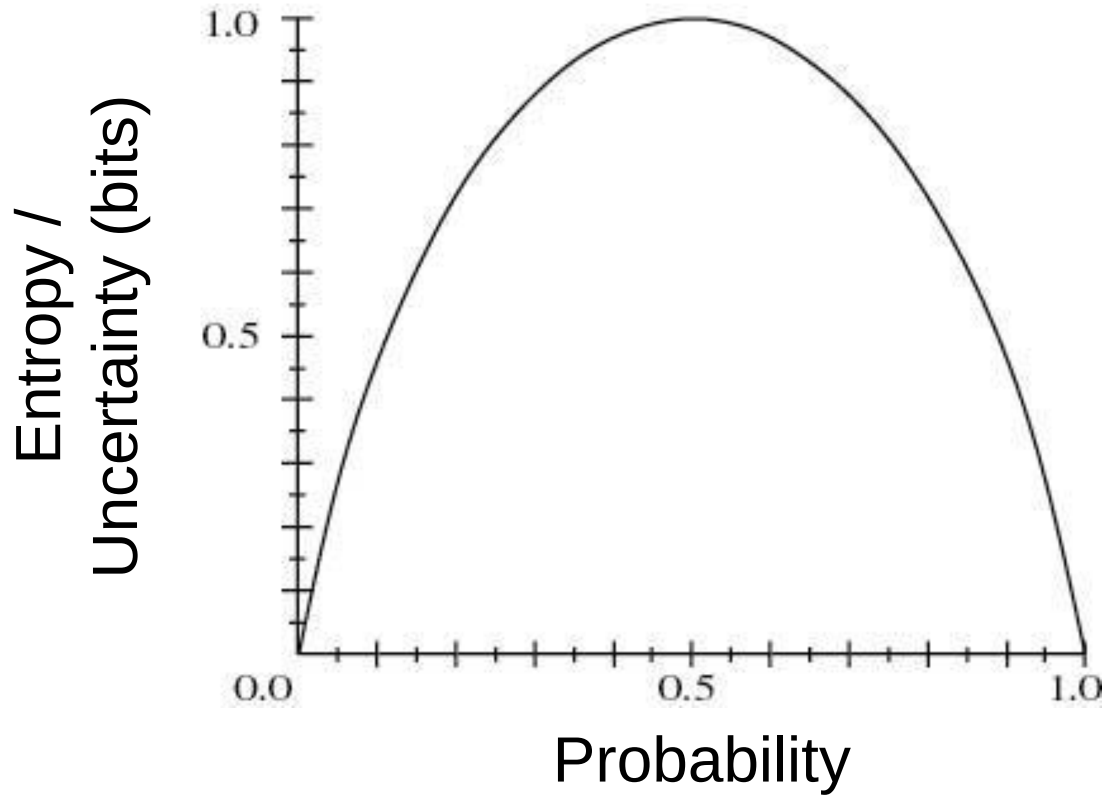
Computing “Information”

- Information is measured in bits.
- Entropy: degree of impurity in the attribute.
  - information required to predict the class, given a probability distribution.
  - $\text{entropy}(p_1,p_2,\ldots,p_n) = -p_1 * \log_2(p_1) - p_2 * \log_2(p_2) - \ldots - p_n * \log_2(p_n)$
    - p_i is the proportion in class i
- Information via entropy of class distribution
  - $\text{info}([c_1,c_2,\ldots,c_n]) = \quad \text{entropy}(c_1/c\ ,\ c_2/c\ ,\ \ldots\ ,\ c_n/c)$
    - c_i is the number of instances of class i
    - $c = c_1+c_2+\ldots+c_n$

Characteristics of Entropy

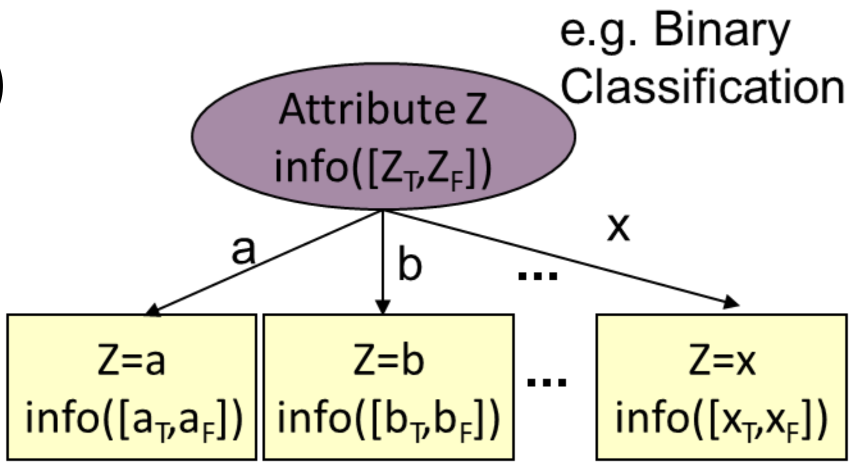
- Entropy is 0
  - If all examples in each branch have the same class (pure node)
    - A class can be attached to the branch, classifying all instances.
    - E.g. entropy(0, 1) = 0
- Entropy is maximal
  - If the examples in each branch are evenly distributed between classes.
    - Most effort is required to differentiate between classes
    - E.g. entropy(0.5, 0.5) = 1

Characteristics of Entropy (continued)



Computing Information Gain

- Expected Information from attribute splitting
  - weighted average of information (entropy) from each partition
- Information Gain
  - difference between
    - information contained in set (before splitting)
    - expected information from attribute split



ROBERT GORDON UNIVERSITY ABERDEEN

Example: weather data

#	Outlook	Temperature	Humidity	Windy	Play
1	Sunny	Hot	High	False	No
2	Sunny	Hot	High	True	No
3	Cloudy	Hot	High	False	Yes
4	Rainy	Mild	High	False	Yes
5	Rainy	Cool	Normal	False	Yes
6	Rainy	Cool	Normal	True	No
7	Cloudy	Cool	Normal	True	Yes
8	Sunny	Mild	High	False	No
9	Sunny	Cool	Normal	False	Yes
10	Rainy	Mild	Normal	False	Yes
11	Sunny	Mild	Normal	True	Yes
12	Cloudy	Mild	High	True	Yes
13	Cloudy	Hot	Normal	False	Yes
14	Rainy	Mild	High	True	No

Example: dataset information

- 14 weather instances
  - 9 Yes and 5 No
- info(data)

  = info([9, 5])

  = entropy(9/14, 5/14)

  = -9/14 $\log_2$ (9/14) – 5/14 $\log_2$ (5/14)

  = 0.940 bits

$\log_2 0.64 = -0.644$

$\log_2 0.36 = -1.474$

Dataset

Examples

Y: 3,4,5,7,9,10,11,12,13

N: 1,2,6,8,14

info([***9***,***5***])

#	Outlook	Temperature	Humidity	Windy	Play
1	Sunny	Hot	High	False	No
2	Sunny	Hot	High	True	No
3	Cloudy	Hot	High	False	Yes
4	Rainy	Mild	High	False	Yes
5	Rainy	Cool	Normal	False	Yes
6	Rainy	Cool	Normal	True	No
7	Cloudy	Cool	Normal	True	Yes
8	Sunny	Mild	High	False	No
9	Sunny	Cool	Normal	False	Yes
10	Rainy	Mild	Normal	False	Yes
11	Sunny	Mild	Normal	True	Yes
12	Cloudy	Mild	High	True	Yes
13	Cloudy	Hot	Normal	False	Yes
14	Rainy	Mild	High	True	No

Note : (log x) / (log 2) = $\log_2$ x

ROBERT GORDON UNIVERSITY ABERDEEN

Table of *log*$_2$ function (also supplied separately as pdf file)

X	*log*$_2$*x*
0.01	-6.644
0.02	-5.644
0.03	-5.059
0.04	-4.644
0.05	-4.322
0.06	-4.059
0.07	-3.837
0.08	-3.644
0.09	-3.474
0.1	-3.322
0.11	-3.184
0.12	-3.059
0.13	-2.943
0.14	-2.837
0.15	-2.737
0.16	-2.644
0.17	-2.556
0.18	-2.474
0.19	-2.396
0.2	-2.322

x	*log*$_2$*x*
0.21	-2.252
0.22	-2.184
0.23	-2.120
0.24	-2.059
0.25	-2.000
0.26	-1.943
0.27	-1.889
0.28	-1.837
0.29	-1.786
0.3	-1.737
0.31	-1.690
0.32	-1.644
0.33	-1.599
0.34	-1.556
0.35	-1.515
0.36	-1.474
0.37	-1.434
0.38	-1.396
0.39	-1.358
0.4	-1.322

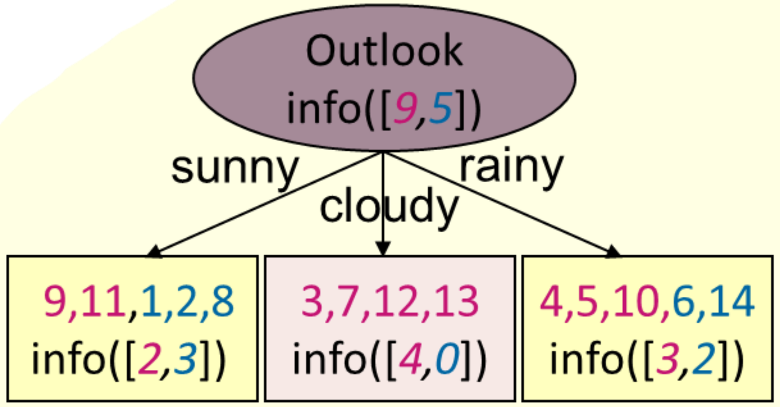
x	*Log*$_2$*x*
0.41	-1.286
0.42	-1.252
0.43	-1.218
0.44	-1.184
0.45	-1.152
0.46	-1.120
0.47	-1.089
0.48	-1.059
0.49	-1.029
0.5	-1.000
0.51	-0.971
0.52	-0.943
0.53	-0.916
0.54	-0.889
0.55	-0.862
0.56	-0.837
0.57	-0.811
0.58	-0.786
0.59	-0.761
0.6	-0.737

x	*log*$_2$*x*
0.61	-0.713
0.62	-0.690
0.63	-0.667
0.64	-0.644
0.65	-0.621
0.66	-0.599
0.67	-0.578
0.68	-0.556
0.69	-0.535
0.7	-0.515
0.71	-0.494
0.72	-0.474
0.73	-0.454
0.74	-0.434
0.75	-0.415
0.76	-0.396
0.77	-0.377
0.78	-0.358
0.79	-0.340
0.8	-0.322

x	*log*$_2$*x*
0.81	-0.304
0.82	-0.286
0.83	-0.269
0.84	-0.252
0.85	-0.234
0.86	-0.218
0.87	-0.201
0.88	-0.184
0.89	-0.168
0.9	-0.152
0.91	-0.136
0.92	-0.120
0.93	-0.105
0.94	-0.089
0.95	-0.074
0.96	-0.059
0.97	-0.044
0.98	-0.029
0.99	-0.014
1	0.000

Example: outlook information

- info(outlook=sunny)
  = info([2,3])
  = $-2/5 \log_2(2/5) - 3/5 \log_2(3/5)$
  = 0.971 bits
- info(outlook=cloudy)
  = info([4,0])
  = $-4/4 \log_2 4/4 - 0/4 \log_2 0/4$
  = 0 bits
- info(outlook=rainy)
  = info([3,2])
  = $-3/5 \log_2(3/5) - 2/5 \log_2(2/5)$
  = 0.971 bits



#	Outlook	Temperature	Humidity	Windy	Play
1	Sunny	Hot	High	False	No
2	Sunny	Hot	High	True	No
3	Cloudy	Hot	High	False	Yes
4	Rainy	Mild	High	False	Yes
5	Rainy	Cool	Normal	False	Yes
6	Rainy	Cool	Normal	True	No
7	Cloudy	Cool	Normal	True	Yes
8	Sunny	Mild	High	False	No
9	Sunny	Cool	Normal	False	Yes
10	Rainy	Mild	Normal	False	Yes
11	Sunny	Mild	Normal	True	Yes
12	Cloudy	Mild	High	True	Yes
13	Cloudy	Hot	Normal	False	Yes
14	Rainy	Mild	High	True	No

Example: information gain

- Weighted average

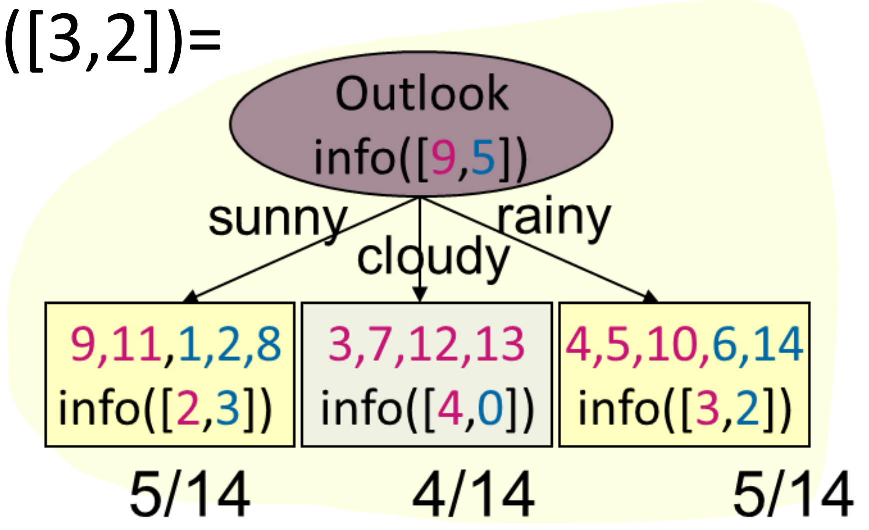
info(outlook) = info([2,3],[4,0],[3,2]) =

5/14*info([2,3]) + 4/14*info([4,0]) + 5/14*info([3,2])=

5/14*0.971 + 4/14*0 + 5/14*0.971 = 0.693

- gain(outlook) = info(data) – info(outlook) =

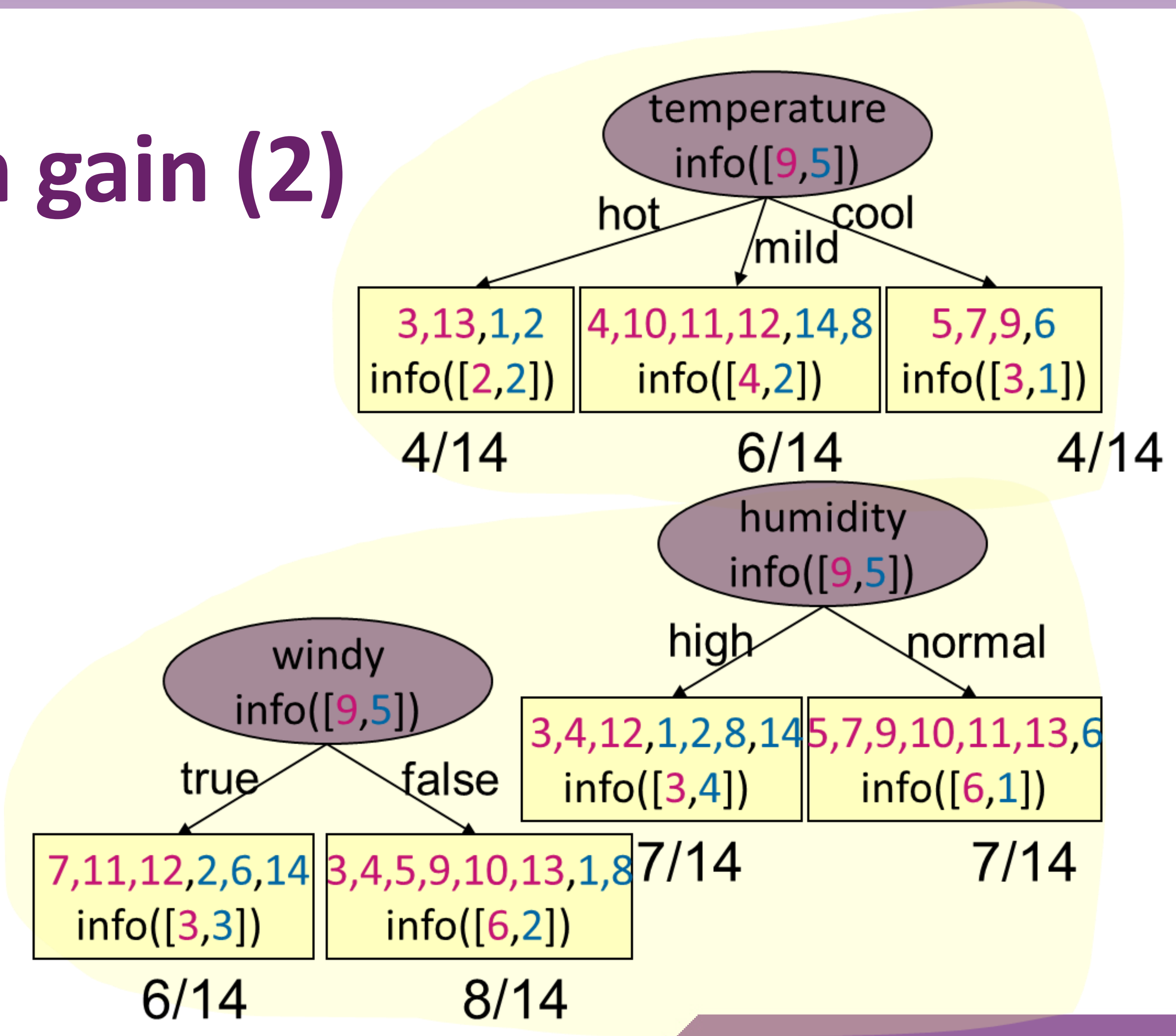
0.940 – 0.693 = 0.247 bits



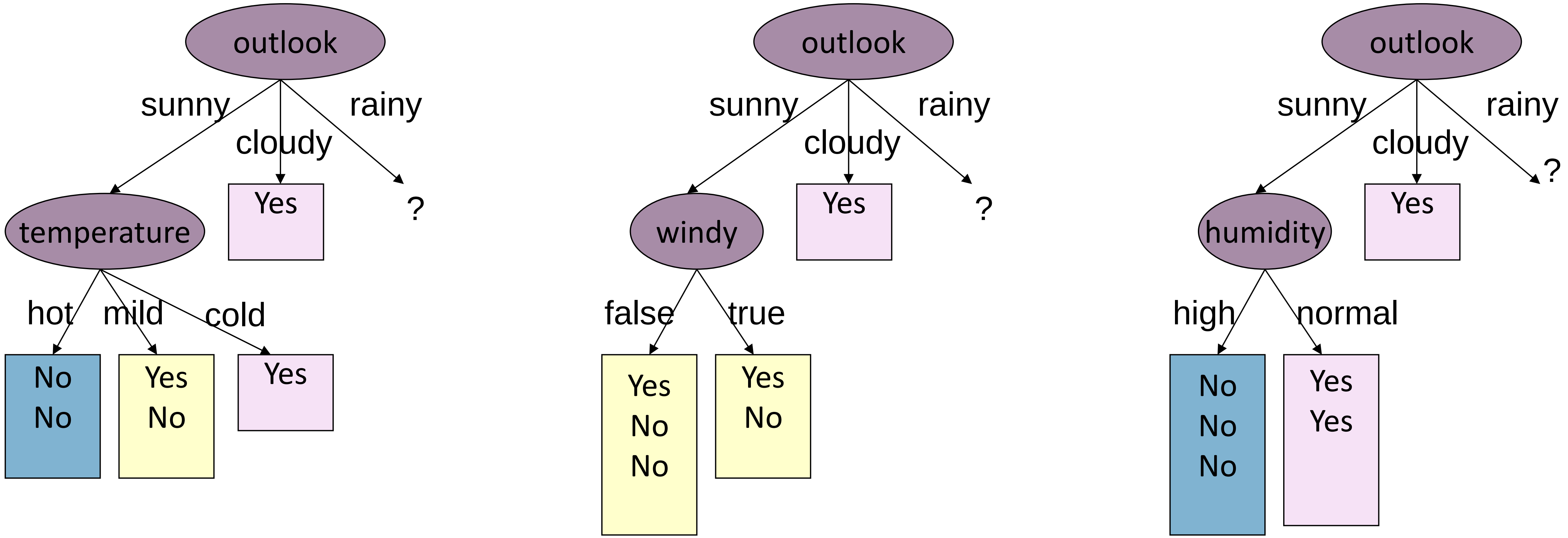
Example: information gain (2)

- Similarly for other attributes
  - gain(temperature) = 0.029 bits
  - gain(humidity) = 0.152 bits
  - gain(windy) = 0.048 bits

- Outlook gives maximum gain
  - gain(outlook) = 0.247 bits
  - Choose outlook as root

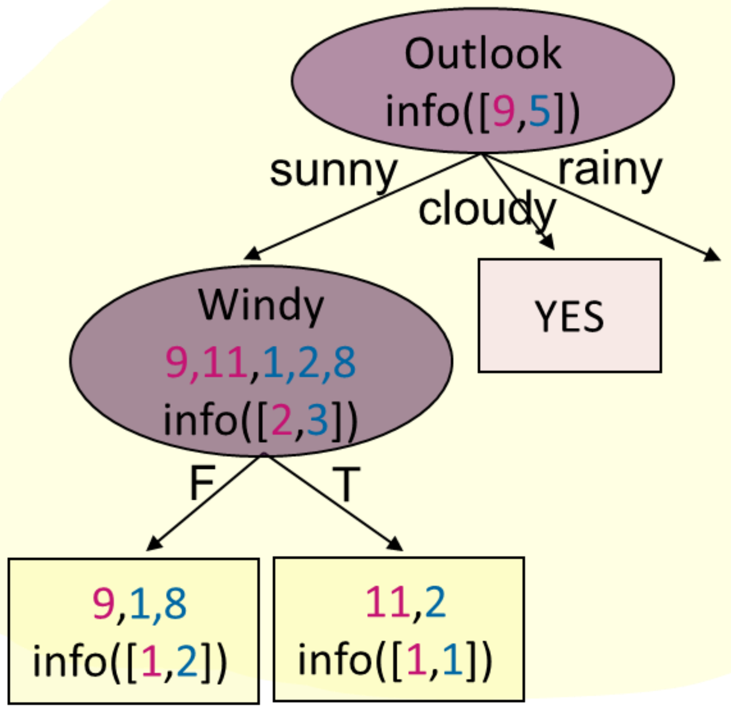


Example: subtrees



Example: sunny subtree

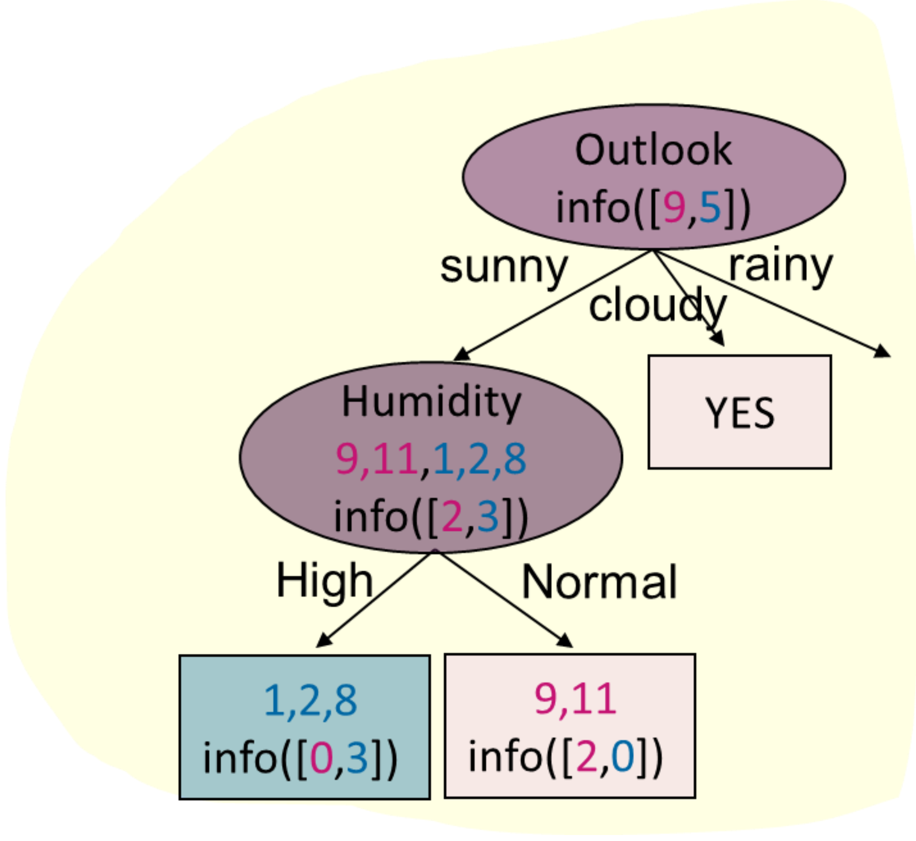
- Sunny data: 5 instances (2 Yes and 3 No)
  - info(sunny data) = 0.971 bits
  - Windy as node
    - info(windy=F) = info([1,2])
      $= -1/3 \log_2(1/3) \ -2/3\log_2(2/3) = 0.91$ bits
    - info(windy=T) = info([1,1])
      $= -1/2 \log_2(1/2) - 1/2 \log_2(1/2) = 1$ bit
  - Expected information for windy
    - info(windy) = info([1,2], [1,1]) = 3/5*0.91 +2/5 * 1 = 0.946 bits
  - Gain(windy) = info(sunny data) – info(windy)
    = 0.971 – 0.946 = 0.025 bits



#	Outlook	Temperature	Humidity	Windy	Play
1	Sunny	Hot	High	False	No
2	Sunny	Hot	High	True	No
8	Sunny	Mild	High	False	No
9	Sunny	Cool	Normal	False	Yes
11	Sunny	Mild	Normal	True	Yes

Example: sunny subtree (continued)

- Sunny data: 5 instances (2 Yes and 3 No)
  - info(sunny data) = 0.971 bits
  - Humidity as node
    - info(humidity=high) = info([0,3]) = $-0/3 \log_2(0/3) - 3/3 \log_2(3/3) = 0$ bits
    - info(humidity=normal) = info([2,0]) = $-2/2 \log_2(2/2) - 0/2 \log_2(0/2) = 0$ bits
  - Expected information for windy
    - info(humidity) = info([3,3], [2,2]) = 3/5*0 +2/5 * 0 = 0 bits
  - Gain(humidity) = info(sunny data) – info(humidity) = 0.971 – 0 = 0.971 bits



#	Outlook	Temperature	Humidity	Windy	Play
1	Sunny	Hot	High	False	No
2	Sunny	Hot	High	True	No
8	Sunny	Mild	High	False	No
9	Sunny	Cool	Normal	False	Yes
11	Sunny	Mild	Normal	True	Yes

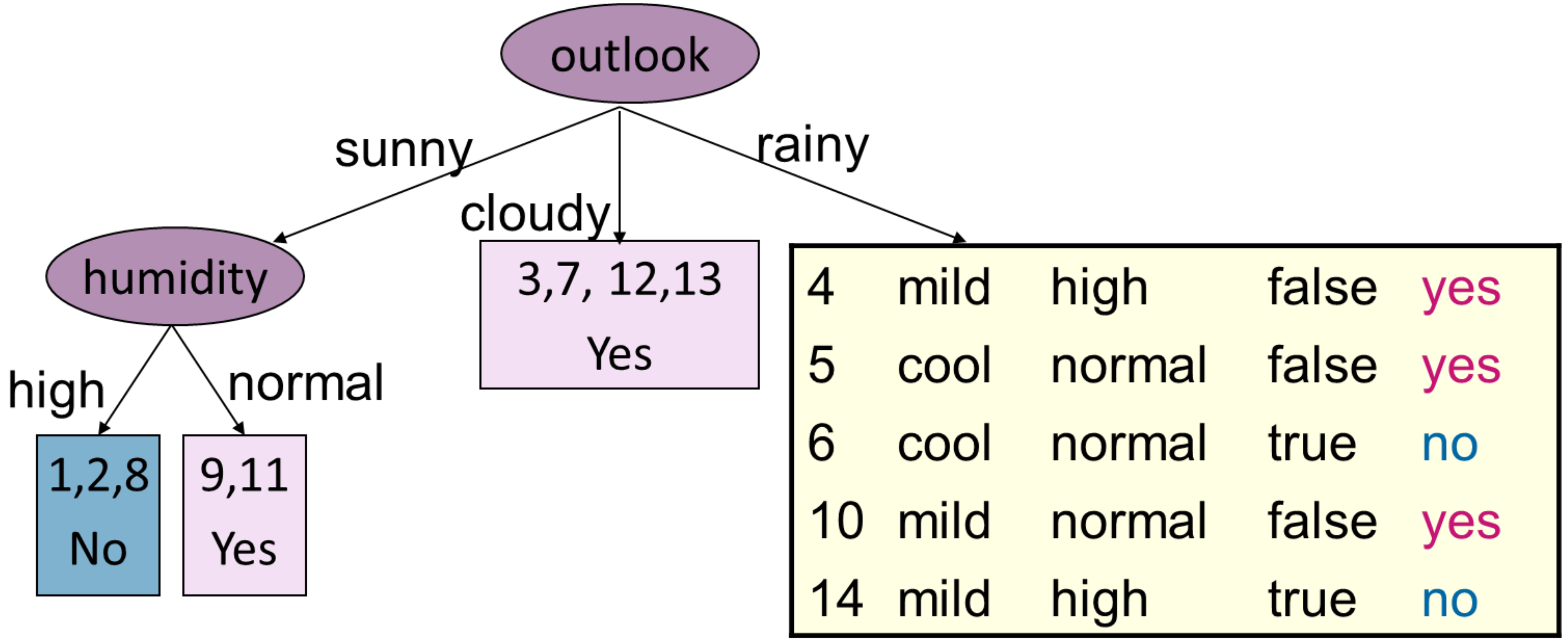
Example: information gain (3)

- Sunny data: 5 instances (2 Yes and 3 No)
  - Gain(temperature) = 0.571 bits
  - Gain(windy) = 0.029 bits
  - Gain(humidity) = 0.971 bits
- Choose humidity as root of outlook=sunny subtree
  - Highest gain

ROBERT GORDON UNIVERSITY ABERDEEN

Selecting node for rainy

#	Outlook	Temperature	Humidity	Windy	Play
4	Rainy	Mild	High	False	Yes
5	Rainy	Cool	Normal	False	Yes
6	Rainy	Cool	Normal	True	No
10	Rainy	Mild	Normal	False	Yes
14	Rainy	Mild	High	True	No



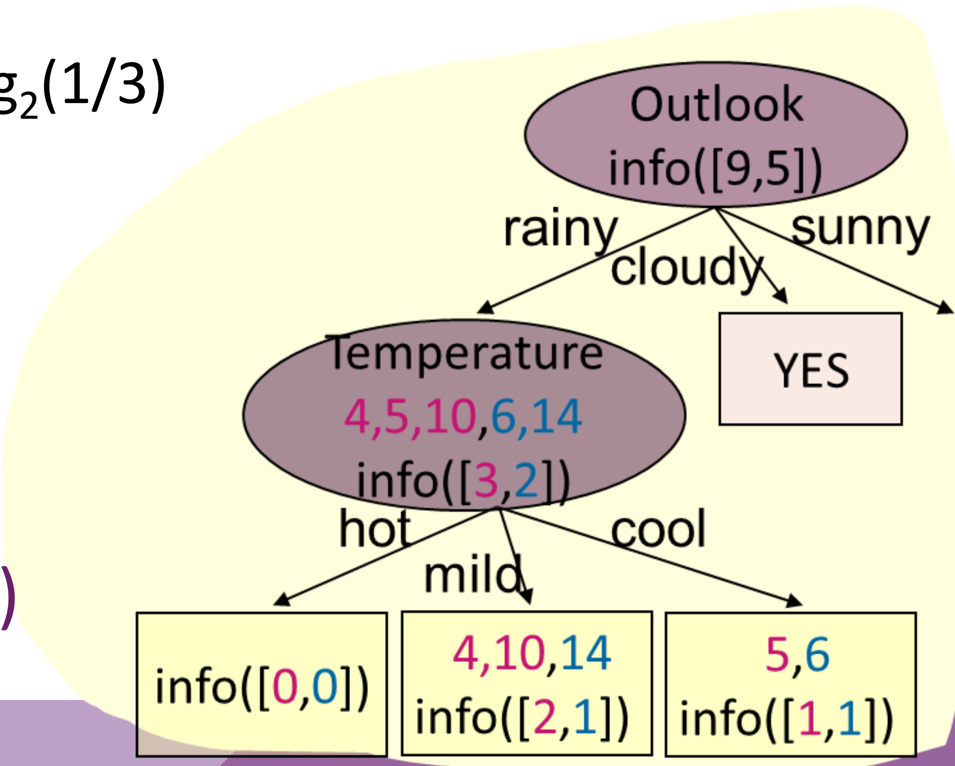
4	mild	high	false	yes
5	cool	normal	false	yes
6	cool	normal	true	no
10	mild	normal	false	yes
14	mild	high	true	no

ROBERT GORDON UNIVERSITY ABERDEEN

Example: rainy subtree

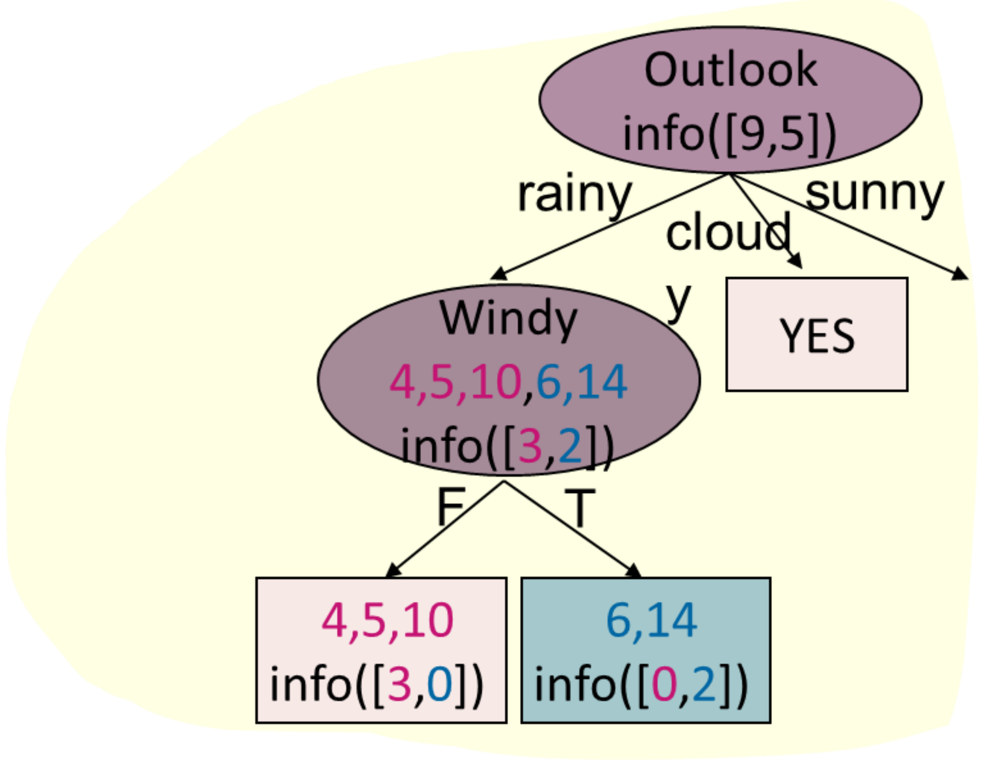
#	Outlook	Temperature	Humidity	Windy	Play
4	Rainy	Mild	High	False	Yes
5	Rainy	Cool	Normal	False	Yes
6	Rainy	Cool	Normal	True	No
10	Rainy	Mild	Normal	False	Yes
14	Rainy	Mild	High	True	No

- Rainy data: 5 instances (3 Yes and 2 No)
  - info(rainy data) = 0.971 bits
  - Temperature as node
    - info(temp=hot) = info([0,0]) = 0 bits
    - info(temp=mild) = info([2,1]) = -2/3 $\log_2(2/3)$ – 1/3 $\log_2(1/3)$ = 0.903 bits
    - info(temp=cool) = info([1,1]) = -1/1 $\log_2(1/1)$ – 1/1 $\log_2(1/1)$= 1 bit
  - Expected information for temperature
    - info(temp) = info([0,0], [2,1], [1,1])
      = 0/5 * 0 + 3/5 * 0.903 + 2/5 * 1 = 0.942 bits
  - Gain(temperature) = info(rainy data) – info(temp)
    = 0.971 – 0.942 = 0.029 bits



Example: rainy subtree (2)

- Rainy data: 5 instances (3 Yes and 2 No)
  - info(rainy data) = 0.971
  - Windy as node
    - info(windy=F) = info([3,0])
      $= -3/3 \log_2(3/3) – 0/3 \log_2(0/3) = 0$ bits
    - info(windy=T) = info([0,2])
      $= -0/2 \log_2(0/2) – 2/2 \log_2(2/2) = 0$ bits
  - Expected information for windy
    - info(windy) = info([3,0], [0,2]) = 3/5 * 0 +2/5 * 0 = 0 bits
  - Gain(windy) = info(rainy data) – info(windy)
    = 0.971 – 0 = 0.971 bits



#	Outlook	Temperature	Humidity	Windy	Play
4	Rainy	Mild	High	False	Yes
5	Rainy	Cool	Normal	False	Yes
6	Rainy	Cool	Normal	True	No
10	Rainy	Mild	Normal	False	Yes
14	Rainy	Mild	High	True	No

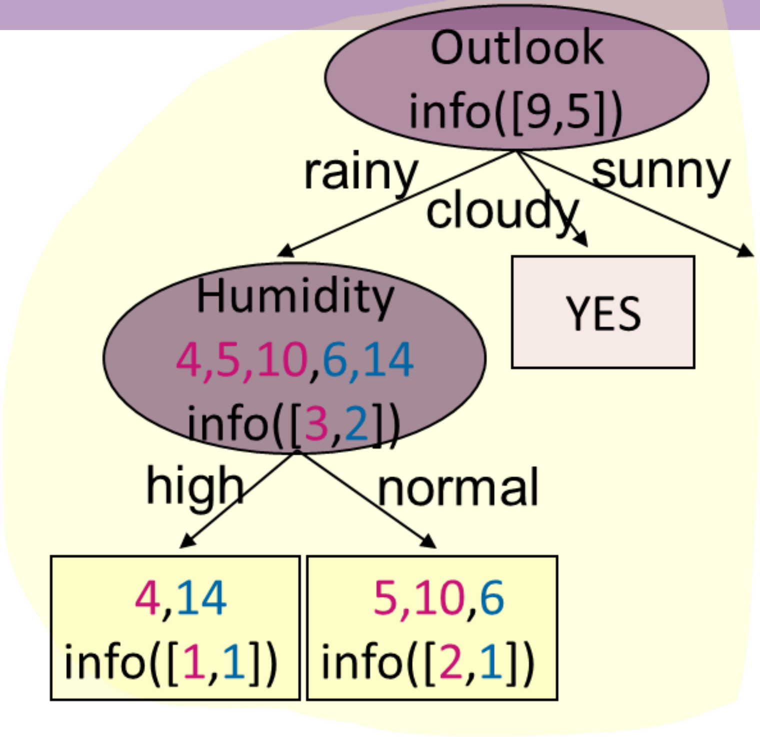
ROBERT GORDON UNIVERSITY ABERDEEN

Example: rainy subtree (3)

- Rainy data: 5 instances (3 Yes and 2 No)
  - info(rainy data) = 0.971
  - Humidity as node
    - info(humidity=high) = info([1,1]) = $-1/2 \log_2(1/2) – 1/2 \log_2(1/2)$ = 1 bits
    - info(humidity=norm)=info([2,1]) = $-2/3 \log_2(2/3) – 1/3 \log_2(1/3)$ = 0.903 bits
  - Expected information for windy

    info(humidity) = info([1,1], [2,1]) = 2/5 * 1 +3/5 * 0.903 = 0.542 bits
  - Gain(humidity) = info(sunny data) – info(humidity)

    = 0.971 – 0.542 = 0.429 bits



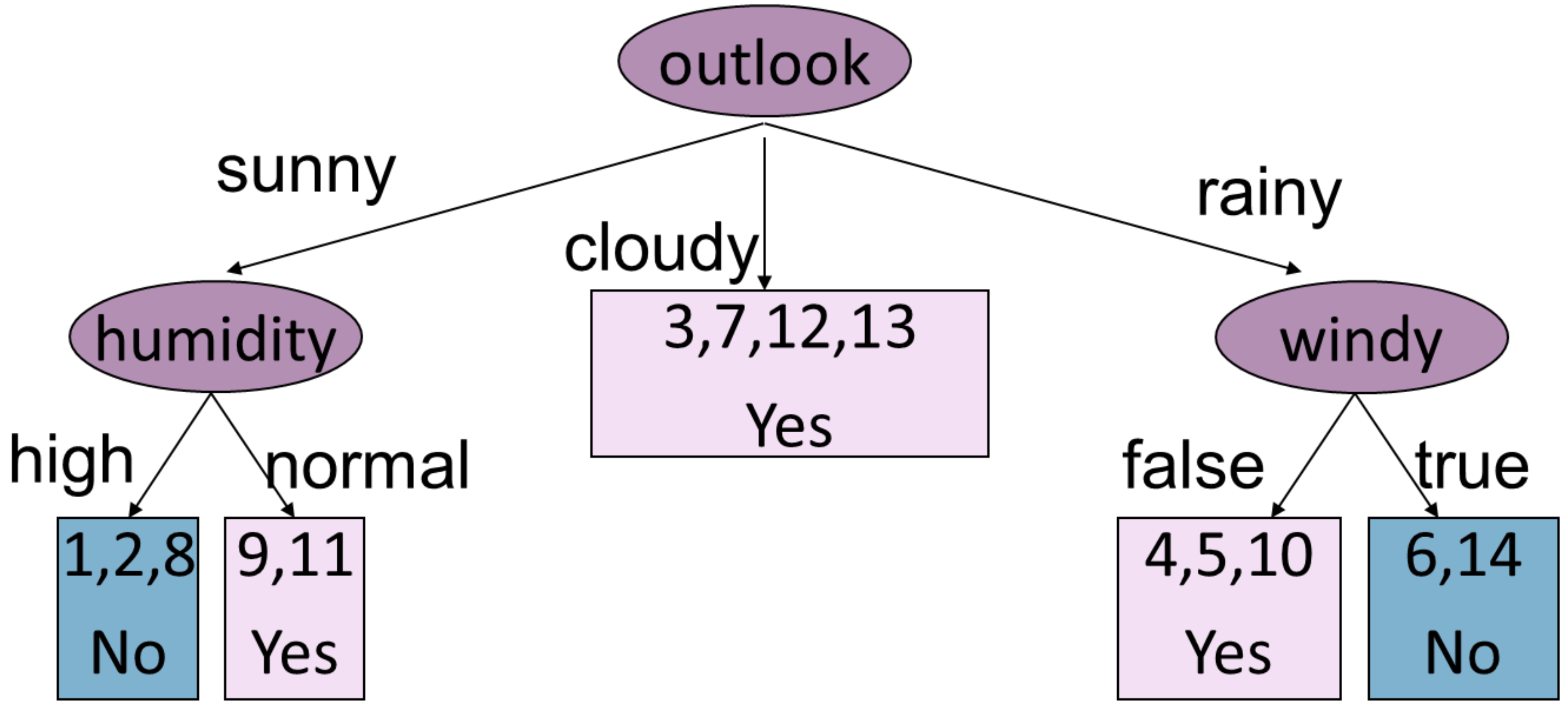
#	Outlook	Temperature	Humidity	Windy	Play
4	Rainy	Mild	High	False	Yes
5	Rainy	Cool	Normal	False	Yes
6	Rainy	Cool	Normal	True	No
10	Rainy	Mild	Normal	False	Yes
14	Rainy	Mild	High	True	No

ROBERT GORDON UNIVERSITY ABERDEEN

Example: information gain for rainy

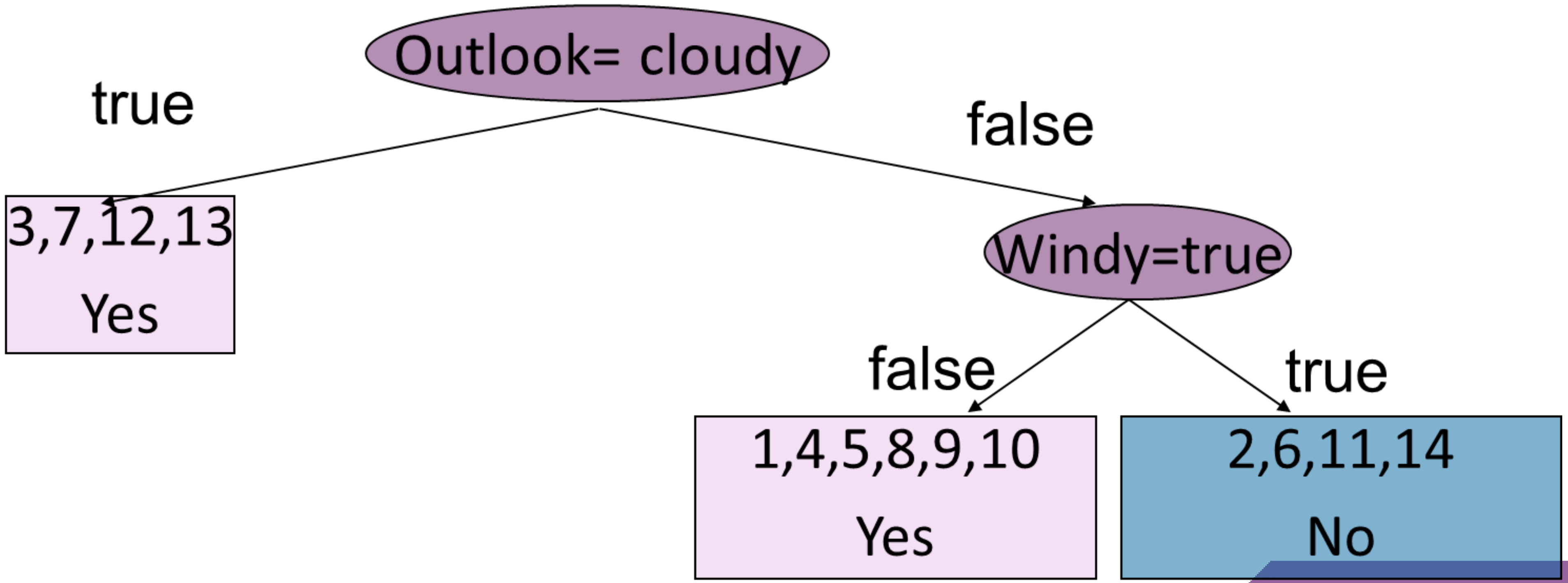
- Rainy data: 5 instances (3 Yes and 2 No)
  - Gain(temperature) = 0.029 bits
  - Gain(windy) = 0.971 bits
  - Gain(humidity) = 0.429 bits
- Choose windy as root of outlook=rainy subtree
  - Highest gain

Solution tree



With binarisation of attributes

- Instead of values being nominal, values may be TRUE (1 or “> 0”) or FALSE (0 or “≤ 0”). For example,



One-hot encoding example

Weather (nominal)	Weather_Sunny	Weather_Rainy	Weather_Cloudy
Sunny	1	0	0
Rainy	0	1	0
Cloudy	0	0	1

Outline

- Pre-processing
- Divide & Conquer Algorithms
- ID3 (original algorithm) , C4.5 (improved algorithm) and C5.0 (commercial version when it came out) use Information Gain
  - Choosing root attribute
  - Choosing attributes for subtrees
- Other methods
  - Gini index - CART
  - Chi-squared - CHAID
- Summary

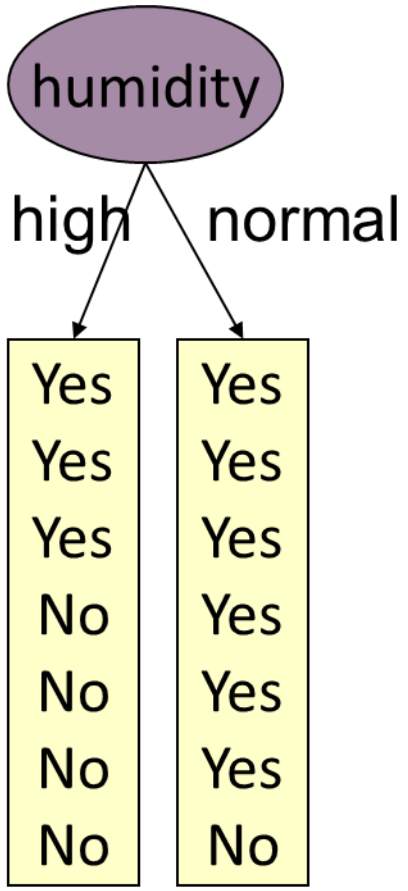
Other algorithms - CART

- CART algorithm uses Gini index with binary attributes
  - Must ***one-hot encode*** attributes *outlook* and *temperature*
    - Include dummy attributes so that all attributes are binary
  - Gini index = 0 if all instances of one class and 0.5 if instances evenly distributed amongst classes
    - Gini (node) = $1 - (p^2 + (1- p)^2)$ where p is the probability of one of the 2 outcomes
  - For a node
    - Calculate Gini index for each sub-node
    - Calculate weighted average of the Gini indexes of the sub-nodes
  - See example below for humidity only
  - Would do for all candidate attributes and select the one with the lowest Gini index.

… other algorithms - CART

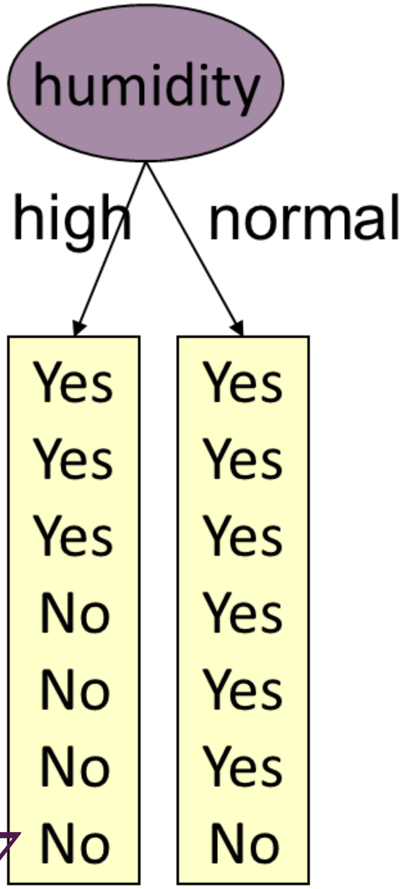
- Partial example with humidity
- For high humidity
  - p (yes) = 3/7 = 0.429
  - Gini (h = high) = 1- $(0.429^2 + (1-0.429)^2)$ =
    1- (0.184 + 0.326) = 0.49
- For normal humidity
  - p (yes) = 6/7 = 0.857
  - Gini (h = normal) = 1- $(0.857^2 + (1-0.857)^2)$ =
    1- (0.734 + 0.02) = 0.246

  Combining for humidity
- Gini (humidity) =7/14 * 0.49 + 7/14 * 0.246 = 0.368
- Calculate for other binary attributes and **choose the one with the lowest Gini index**.



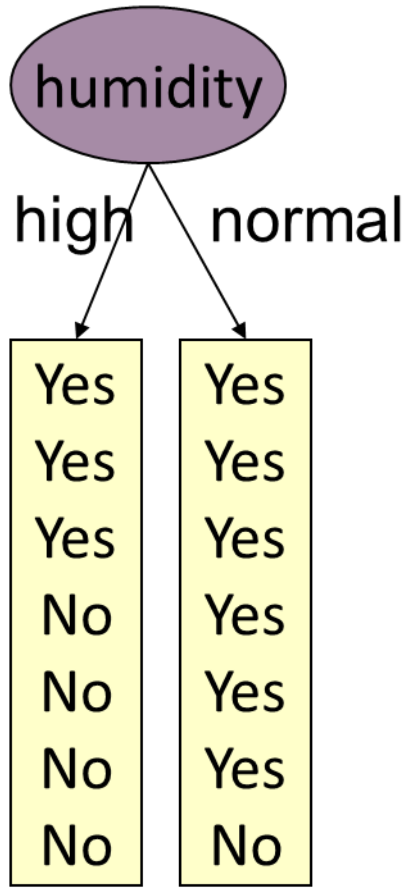
… other - CHAID

- CHAID can also work – class is binary (but can work with non-binary class too).
- $chiSquare(node) = \sqrt{\left(\frac{(actual-expected)^2}{expected}\right)}$
- Process
    - Calculate chi-Square for each class in each branch
    - Add all values for all branches
- 9 *yes* out of 14 so for each branch expect 4.5 *yes*
- 5 *no* out of 14 so for each branch expect 2.5 *no*
- For humidity = high (h=high)
    - For yes: $chiSquare(yes, h = high) = \sqrt{\left(\frac{(3-4.5)^2}{4.5}\right)} = 0.707$
    - For no: $chiSquare(no, h = high) = \sqrt{\left(\frac{(4-2.5)^2}{2.5}\right)} = 0.949$



… other CHAID

- For humidity - normal
  - For yes: $chiSquare(yes, h = normal) = \sqrt{\left(\frac{(6-4.5)^2}{4.5}\right)} = 0.707$
  - For no: $chiSquare(no, h = normal) = \sqrt{\left(\frac{(1-2.5)^2}{2.5}\right)} = 0.949$
- Overall
  - $sumchiSquare(humidity) = 0.707 + 0.949 + 0.707 + 0.949 = 3.312$
- Do same for other attributes and **choose the one with the highest sum of chi-squares**.



Summary

- Classification trees use
  - “Divide and conquer” approach
  - Models produced are easy to interpret
  - Risk of overfitting
  - Several methods for splitting node into branches
    - Information gain (C5.0)
    - Gini index (CART)
    - Chi square (CHAID)
- Simplicity –
  - Smaller decision trees are better