Decision Trees

Danushka Bollegala

Rule-based Classifiers

- In rule-based learning, the idea is to *learn* a rule from train data in the form IF X THEN Y (or a combination of nested conditions) that explains when Y would be TRUE

- Example
  - IF forecast=rainy THEN play=NO
  - If the weather forecast says it is going to rain, then we will not have a match tomorrow.

Pros/Cons

- Advantages of Rule-based Learners
  - Rules are often easier to understand
    - Compare this with the perceptron's weight vector we studied last week
  - We can handle features (aka attributes) that have categorical values (e.g. forecast=sunny/rainy) without requiring those to be mapped to numerical values
- Disadvantages of Rule-based Learners
  - Tend to overfit to the train data more easily than some of the other classification algorithms (k-NN, SVMs)
  - Rules can get very complicated when we have millions of features
    - Consider learning a rule for sentiment classification from texts
  - Old-school machine learning :-) but there are some modern versions with good performance such as Random Forest Classifiers.

Dataset

outlook	temperature	humidity	windy	play?
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Task

- We would like to learn a rule that predicts when we would play
  - IF ….. THEN play=yes
- Requirements
  - The rule must be accurate (it should correctly predict the training instances as much as possible)
  - The rule must be simple
    - Occam’s razor

Occam's Razor

- If something can be explained by two hypotheses A and B, and if A is simpler than B, then we should *prefer* A over B
- Razor?
  - chops off complicated explanations such as the hypothesis B above
- Why we would care?
  - Complex rules (lots of IF conditions) are likely overfit to the train data, which is bad.

William of Ockham
(1287-1347 Surrey UK)

Dataset Properties

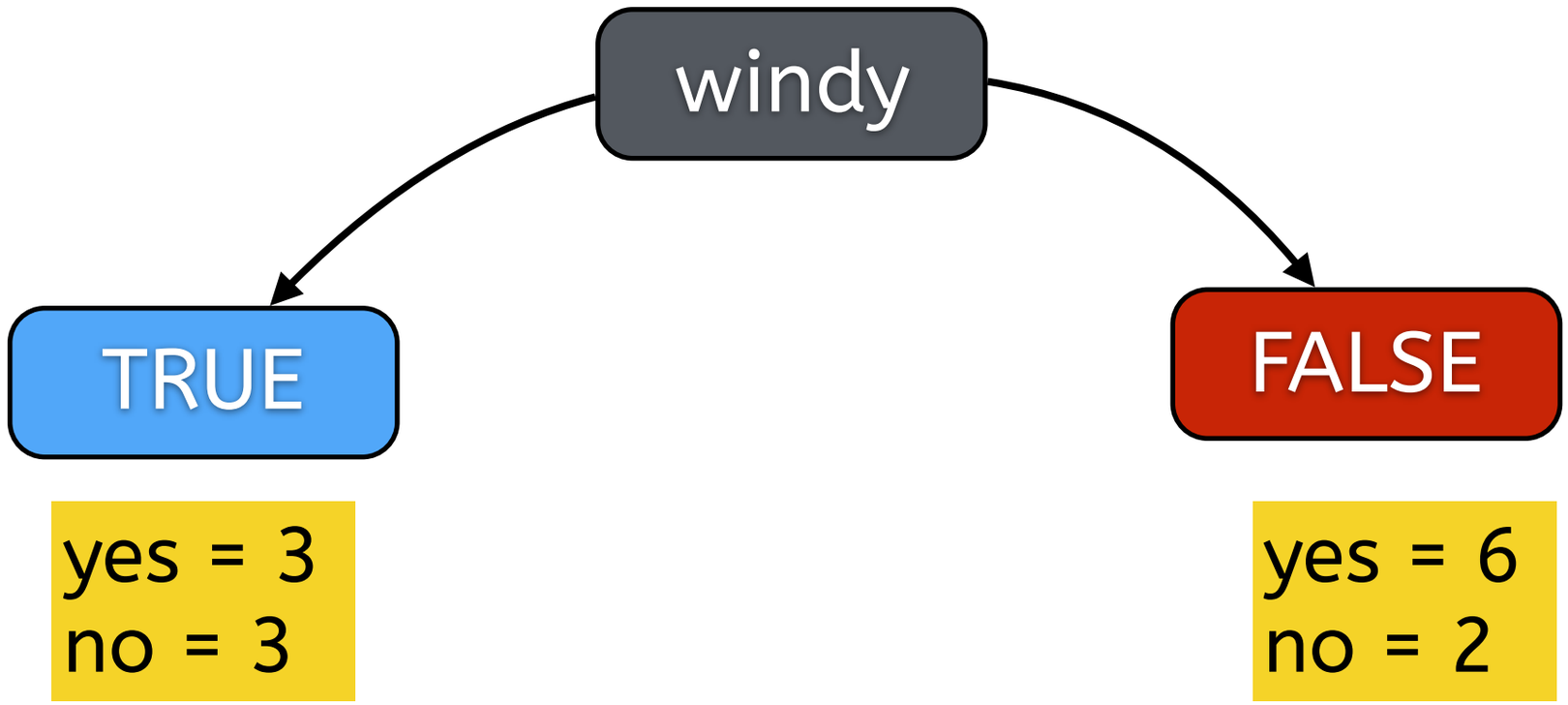
- We have 14 training instances
- We have four features
  - outlook ∈ {sunny, overcast, rainy}
  - temp ∈ {hot, mild, cool}
  - humidity ∈ {high, normal}
  - windy ∈ {FALSE, TRUE}
- Target prediction
  - play ∈ {yes, no}
  - A binary classification task

Tree Learning Algorithm

```
create an empty tree T

select a feature A from the set of features

create branches in T for each value v of A

for each branch

    recurse with instances where A = v

    add tree as branch node
```

How to select a feature to branch?

- Lets make a random guessing algorithm and see how we can improve it
- Let us assume that we selected “windy” as the first feature (selected randomly among the four features in the dataset)



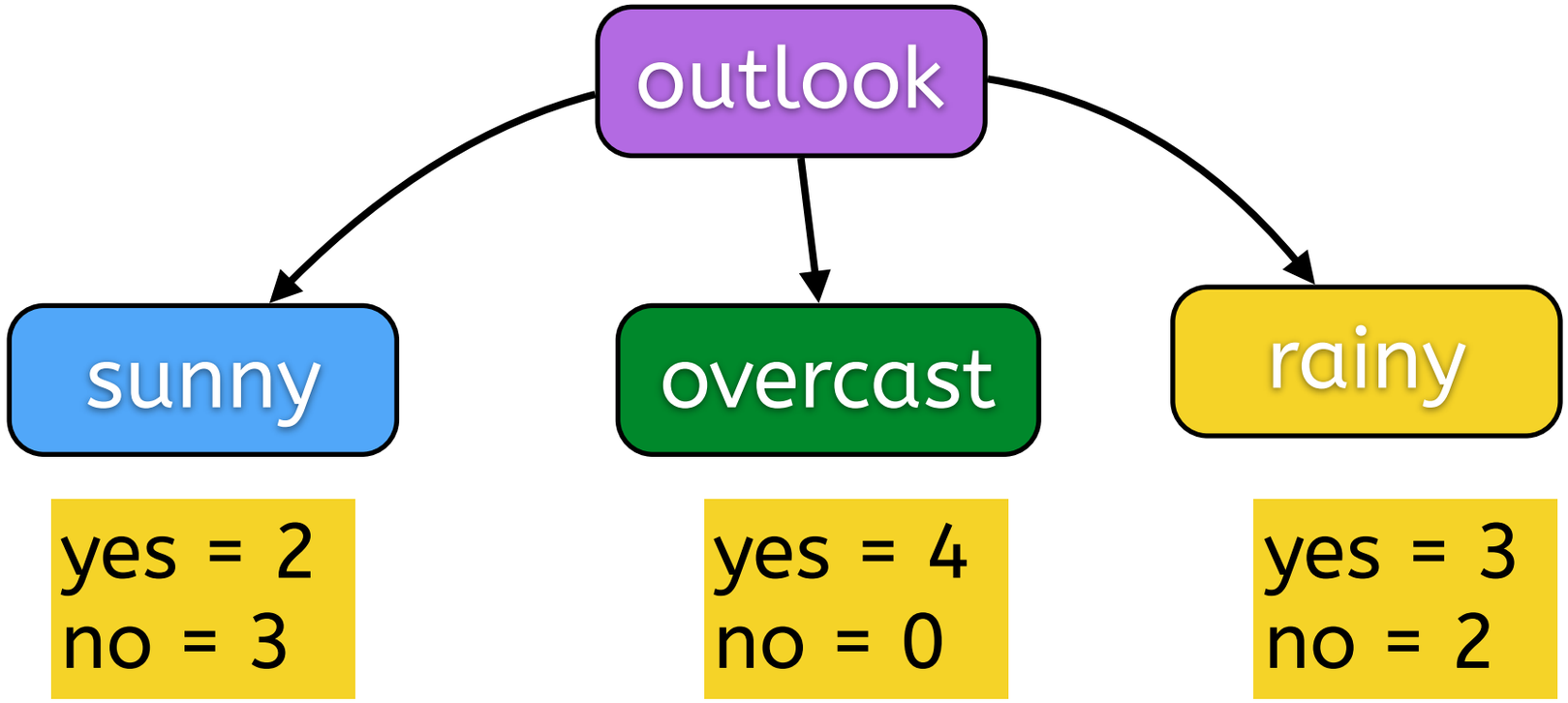
IF windy is TRUE or FALSE, we cannot identify a situation where play will always be yes or no.
Can we find a better feature that splits yes/no “purely”?

Dataset

outlook	temperature	humidity	windy	play?
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Lets make a second random guess

- Let us assume we selected “outlook” as the first feature (selected randomly among the four features in the dataset)



Dataset

outlook	temperature	humidity	windy	play?
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	TRUE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Rule we can learn

- IF outlook = overcast THEN play = yes
- This rule matches 4 out of 14 instances
  - coverage = 4/14 = 0.28
- When it matches, it is perfect!
  - accuracy = 4/4 = 1.0
- But what about the 14-4 = 10 instances for which this rule does not match?
  - This rule cannot be used to classify those 10 instances.
  - We need to look into the other features as well
- Features that give us “pure” splits are better because we can “divide and conquer” using such features more efficiently!

Can we express “purity” empirically?

- (yes=4, no=0) is a more pure split than (yes=2, no=2). Can we make this intuition empirical?
- There are numerous measures that would evaluate how disproportionate a split is. One very popular such measure is the **entropy**

Entropy

- Is a measure of “surprise”
- How surprised will you be if you hear that you got a red ball when you randomly picked a ball from a bag that has 3 red balls and 3 blue balls vs. when you pick a ball from a ball that has 5 red balls and 1 blue ball?
- A measure of the amount of information that we get when we hear someone picked a red ball from the type of bags we discussed above

Entropy— Definition

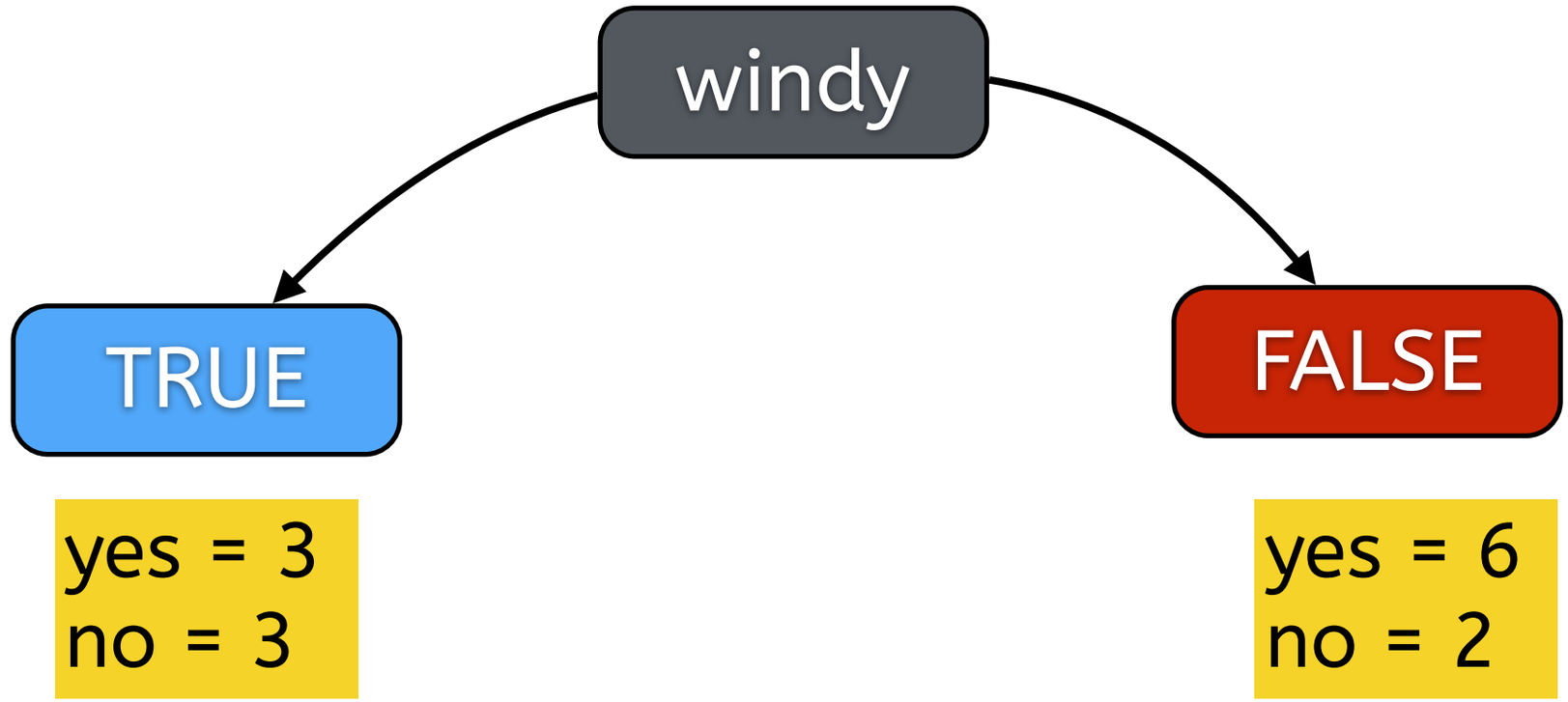
$$H(p) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

- red balls = 3 and blue balls = 3
  - p(red) = 3/6 = 0.5
  - p(blue) = 3/6 = 0.5
  - H(p) = - (0.5 x log(0.5) + 0.5 x log(0.5)) = 1
    - (log(0.5) = -1 for base 2 logarithm)
  - We get 1byte of information (the maximum in this case) when we hear some one picked a red ball from this bag because anything is possible (50-50 chance) and we have no idea before hand what the outcome of this random draw would be.

Back to the example

- Play=no for 5 out of the 14 instances in our dataset.
- Therefore, the original entropy (before we do any branching) is:
- -5/14*log(5/14) - 9/14*log(9/14) = 0.9402

If we select “windy” as the first feature for branching



-3/6*log(3/6)-3/6*log(3/6)
= 1

-6/8*log(6/8)-2/8*log(2/8)
= 0.811

We have 6 instances for windy=TRUE and 8 instances for windy=FALSE. Therefore, the expected entropy is:
6/14 * 1 + 8/14 * 0.811 = 0.8919
Information Gain (IG) = 0.9402 - 0.8919 = 0.0483

Information Gain (IG)

- Information gain (IG) when using a feature f to branch the tree is defined as the **difference between** the amount of entropy we have **before we branch** using f and **after we branch** using f.

- We would prefer f that would yield the maximum information gain

  - If we can gain more information about the dataset using a particular feature f then we will select that feature first for branching

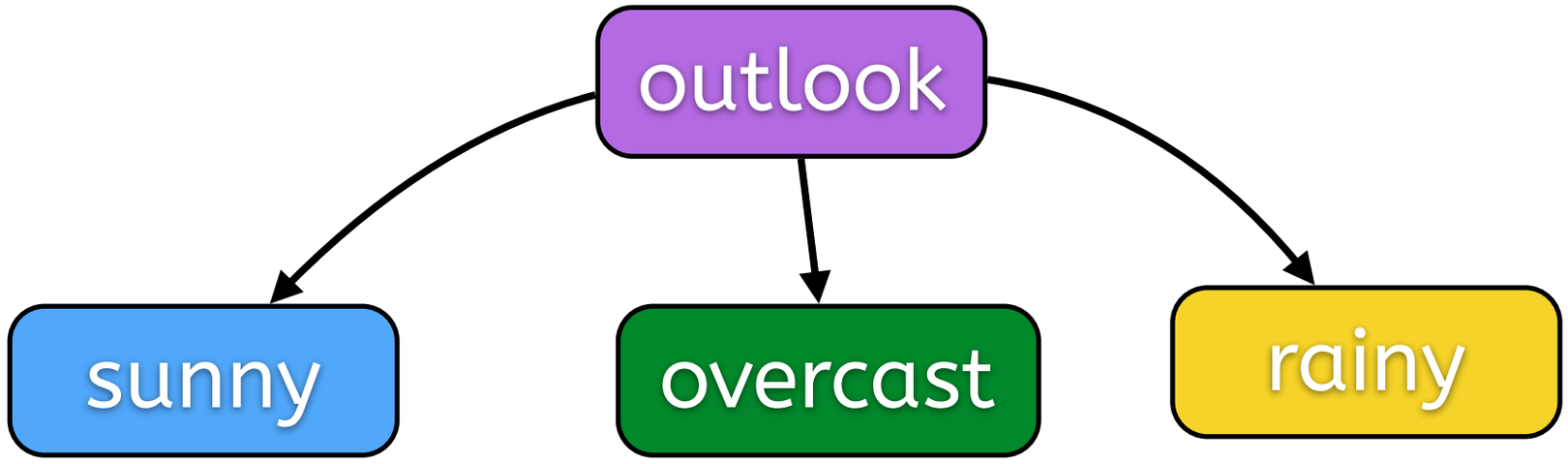
  - A greedy algorithm but works well in practice

Computing Entropy

```
import math

def log2(x):
    return math.log(x) / math.log(2)

def entropy(x, y):
    t = float(x + y)
    e = (x / t) * log2(x / t) + (y / t) * log2(y / t)
    return -e

if __name__ == "__main__":
    print entropy(2,6)
```

On the other hand…

- If we select “outlook” as the first feature



yes = 2
no = 3

yes = 4
no = 0

yes = 3
no = 2

H(sunny)=0.9709

H(overcast) = 0

H(rainy)=0.9709

Expected entropy=5/14*H(sunny)+4/14*H(overcast)+5/14*H(rainy)
= 0.6935
Information Gain = 0.9402 - 0.6935 = 0.2467

Dataset

outlook	temperature	humidity	windy	play?
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	TRUE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Dataset

outlook	temperature	humidity	windy	play?
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	TRUE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

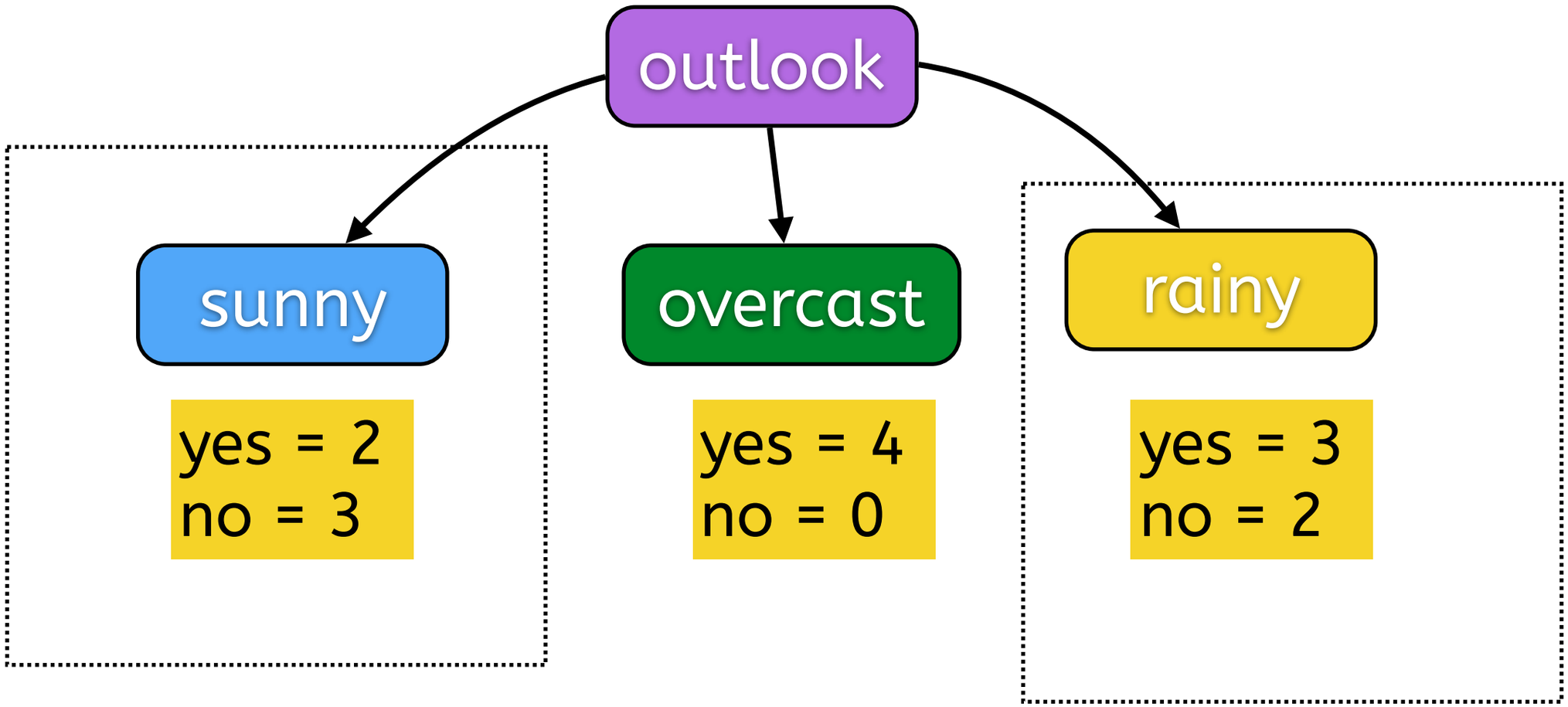
Computing Information Gain (IG)

```
import math

def log2(x):
    return 0 if x == 0 else math.log(x) / math.log(2)

def entropy(x, y):
    t = float(x + y)
    e = (x / t) * log2(x / t) + (y / t) * log2(y / t)
    return -e

def IG(L):
    (xold, yold, tot) = (0, 0, 0)
    for (x, y) in L:
        xold += x
        yold += y
        tot += x + y
    original = entropy(xold, yold)
    evals = [entropy(x,y) for (x,y) in L]
    newval = 0
    for i in range(len(L)):
        newval += (float(sum(L[i])) / float(tot)) * evals[i]
    return original - newval

if __name__ == "__main__":
    print "outlook =", IG([(3,2), (0,4), (2,3)])
    print "windy =", IG([(2,6), (3,3)])
    print "temp =", IG([(2,2), (2,4), (1,3)])
    print "humidity =", IG([(4,3), (1,6)])
```

Computing information gains

- IG(windy) = 0.0483
- IG(outlook) = 0.2467
- IG(temp) = 0.029
- IG(humidity) = 0.1518
- outlook is the clear winner here. By selecting outlook, we maximize the information gain

First branching



We need to further branch the “sunny” and “rainy” sub-trees.

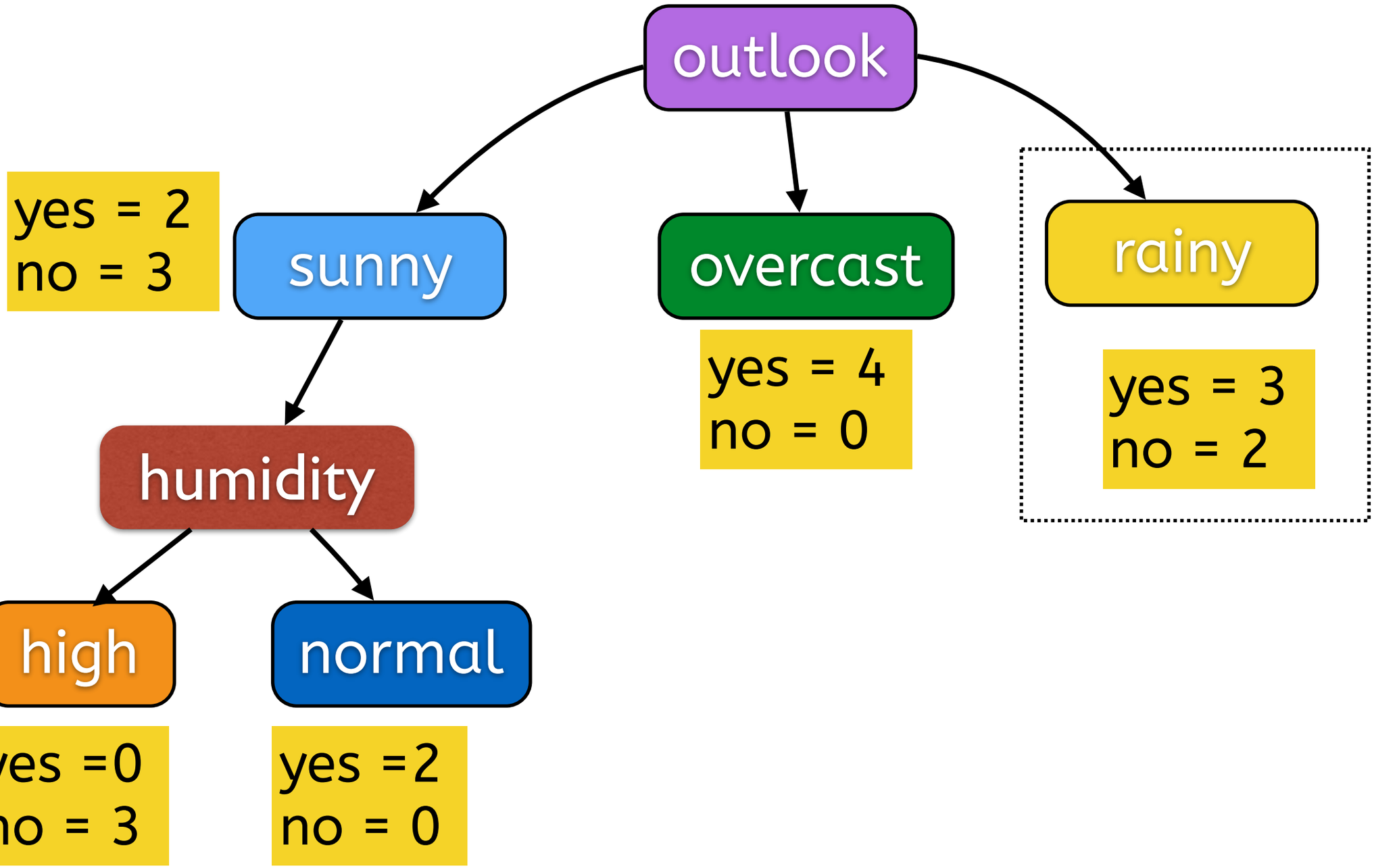
outlook=sunny dataset

outlook	temperature	humidity	windy	play?
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
sunny	mild	normal	TRUE	yes

No need for math here

- IF outlook=sunny AND humidity=high THEN play=no
- But lets do the math anyway…
  - IG(humidity) = 0.9709
  - IG(temp) = 0.5709
  - IG(windy) = 0.0199
- “humidity” wins!

The Decision Tree so far…



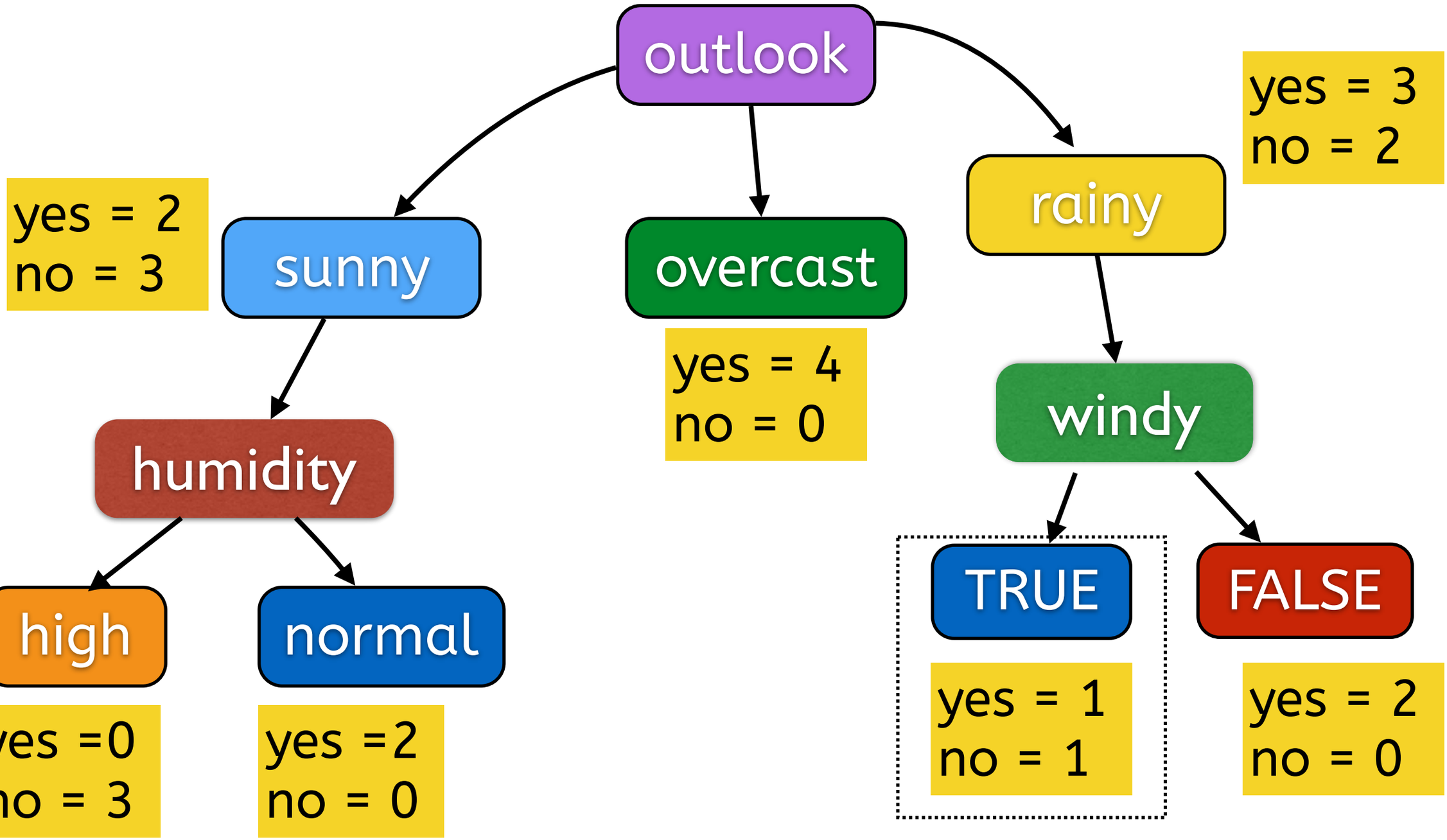
outlook=rainy dataset

outlook	temperature	humidity	windy	play?
rainy	mild	high	FALSE	yes
rainy	cool	normal	TRUE	yes
rainy	cool	normal	TRUE	no
rainy	mild	normal	FALSE	yes
rainy	mild	high	TRUE	no

Computing Information Gains

- IG(windy) = 0.4199
- IG(temp) = 0.0199
- IG(humidity) = 0.0199
- The largest IG is due to “windy”

Final Decision Tree



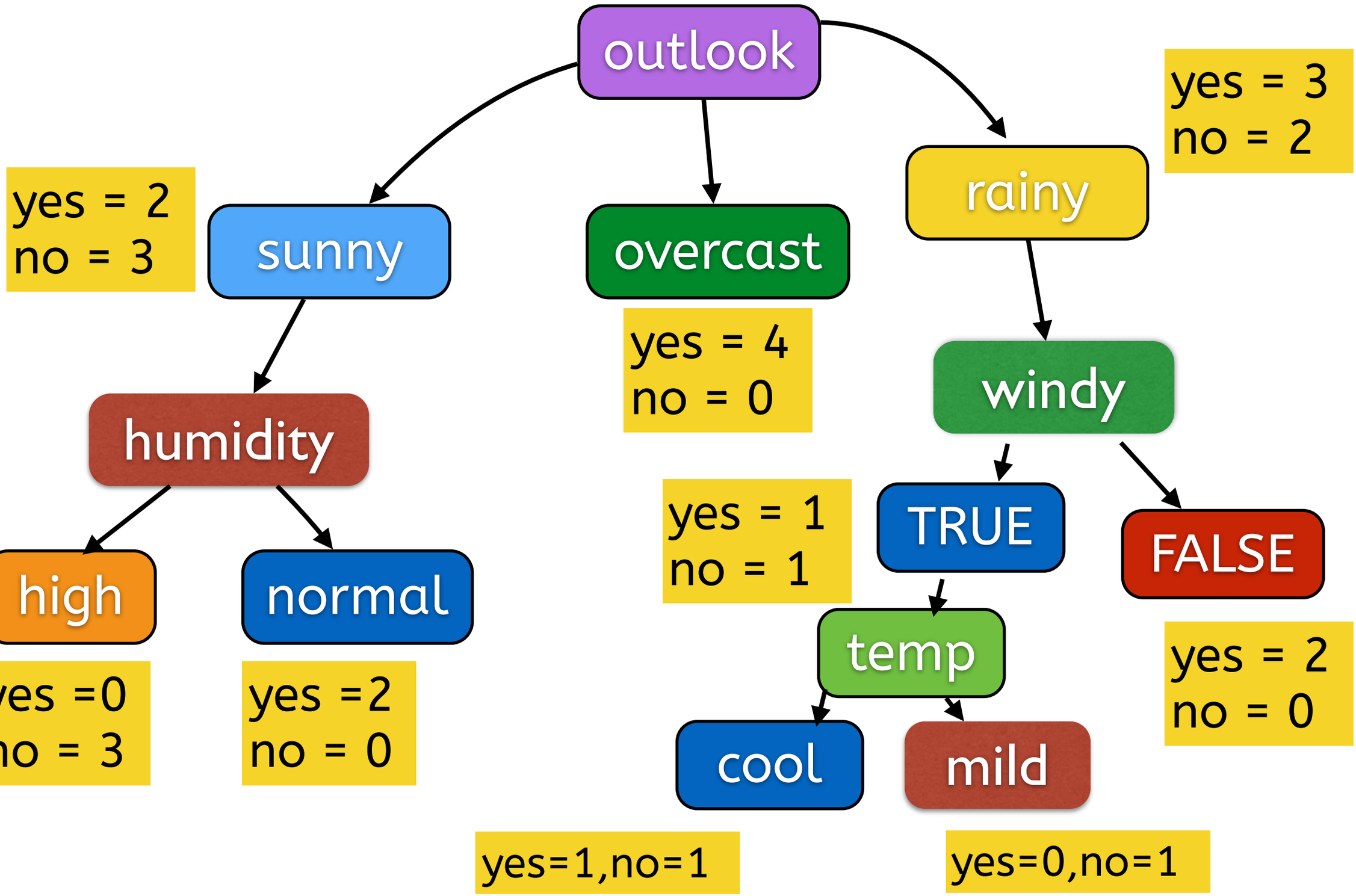
outlook=rainy AND windy=TRUE dataset

outlook	temperature	humidity	windy	play?
rainy	cool	normal	TRUE	yes
rainy	cool	normal	TRUE	no
rainy	mild	high	TRUE	no

Computing Information Gains

- IG(temp) = 0.2516
- IG(humidity) = 0.2516
- No clear winner here. It is a random guess between temp vs. humidity
- Lets select temp

Final Decision Tree



outlook=rainy AND windy=TRUE AND temp = cool

outlook	temperature	humidity	windy	play?
rainy	cool	normal	TRUE	yes
rainy	cool	normal	TRUE	no

Final feature selection

- We are only left with humidity
- But humidity is always “normal” for the remaining instances and we have one “yes” and one “no” for play.
- The dataset cannot be further branched
- All remaining instances (two in total) must be included as exceptions (remembered) to the rule

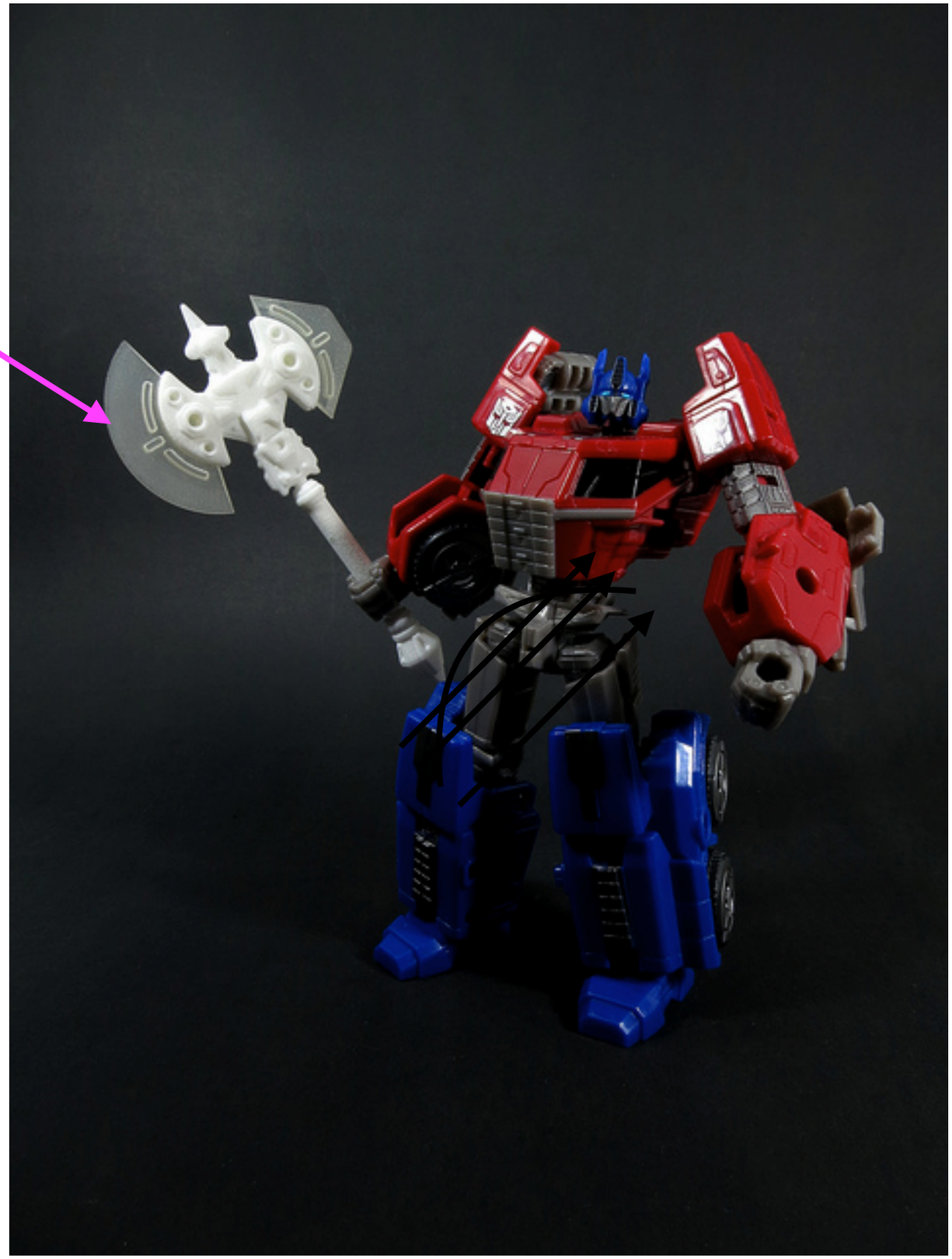
Final Rule

- IF (outlook = overcast) OR
- ((outlook = sunny) AND (humidity = normal)) OR
- ((outlook = rainy) AND (windy = FALSE))
  - THEN play = yes
- ELSE play = no

ID3 Algorithm

- The algorithm that we just learnt is called the ID3 algorithm (Iterative Dichotomizer)
- Proposed by John Ross Quinlan
- J. R. Quinlan, *Induction of Decision Trees*, **Machine Learning**, pp. 81-106, vol. 1, 1986.

Dichtomizer



Decision Tree Issues

- Ordering of Attribute Splits
  - As seen, we need to build the tree picking the best attribute to split first. (greedy)
- Numeric/Missing
  - Dividing numeric data is more complicated. We need to perform some form of a “binnig” (i.e. discretization) as a pre-processing step.
- Tree structure
  - A balanced tree with the fewest levels is preferable.
- Stopping criteria
  - When should we stop (to avoid overfitting)
- Pruning
  - It may be beneficial (speed/over-fitting) to prune the tree once created. We can do this after we have created the tree or while creating the tree (incrementally)

Quiz

- Learn a decision tree that predicts whether a car is fast from the following dataset.

Model	Engine	SC/Turbo	Weight	Fuel Eco	Fast
Prius	small	no	average	good	no
Civic	small	no	light	average	no
WRX STI	small	yes	average	bad	yes
M3	medium	no	heavy	bad	yes
RS4	large	no	average	bad	yes
GTI	medium	no	light	bad	no
XJR	large	yes	heavy	bad	no
S500	large	no	heavy	bad	no
911	medium	yes	light	bad	yes
Corvette	large	no	average	bad	yes
Insight	small	no	light	good	no
RSX	small	no	average	average	no
IS350	medium	no	heavy	bad	no
MR2	small	yes	average	average	no
E320	medium	no	heavy	bad	no