Machine Learning – COMS3007

Decision Trees

Benjamin Rosman
Benjamin.Rosman1@wits.ac.za / benjros@gmail.com

Based heavily on course notes by Chris Williams and Victor Lavrenko, and Clint van Alten

Tennis data

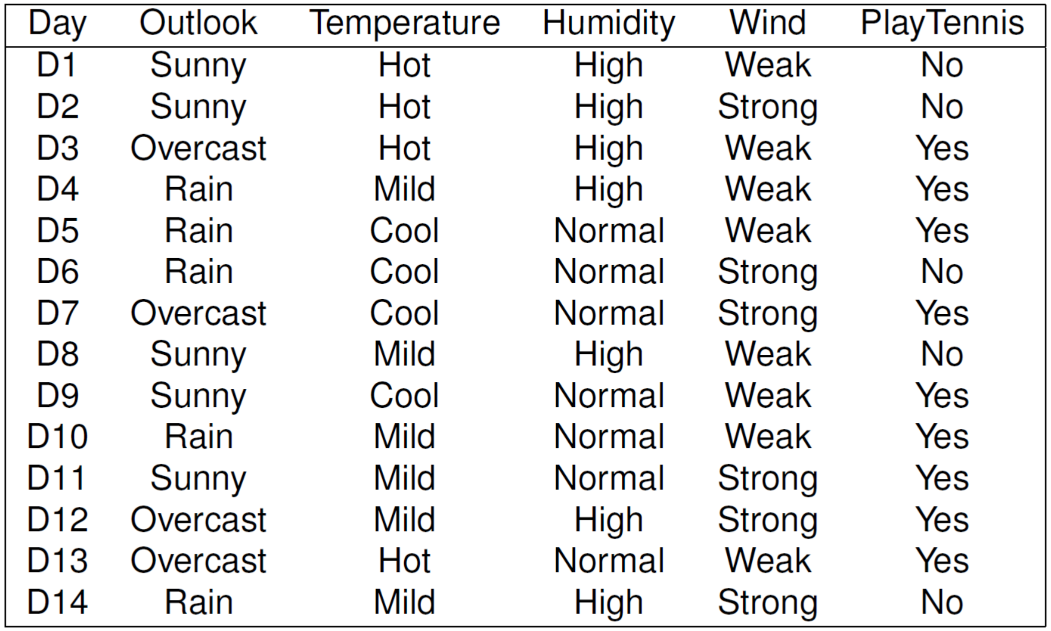
What type of ML is this?

Features

Class

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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

Tennis data



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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, Mild, Normal, Weak)?
- (Sunny, Cool, High, Weak)?
- (Overcast, Hot, Normal, Strong)?

Example – play tennis?

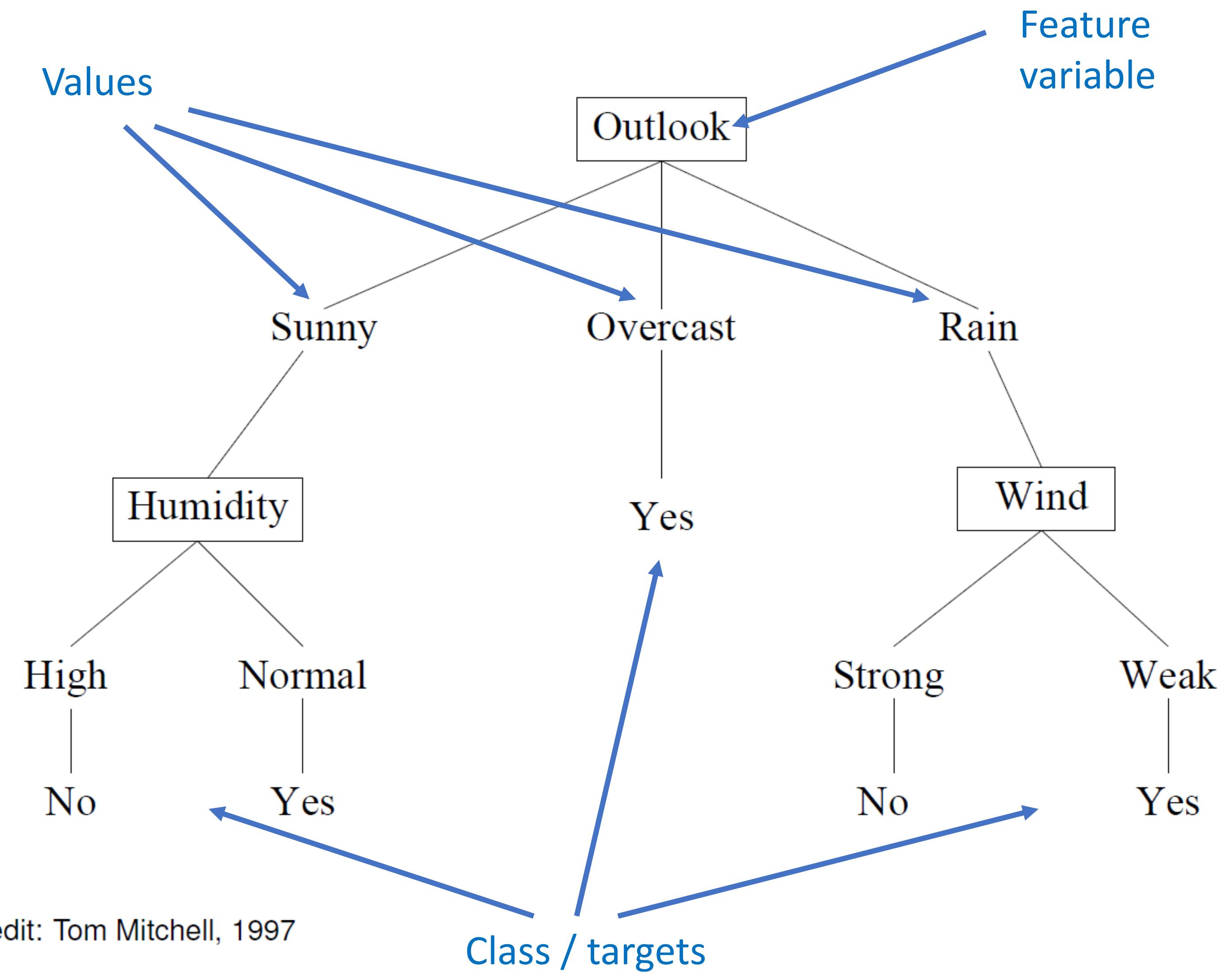


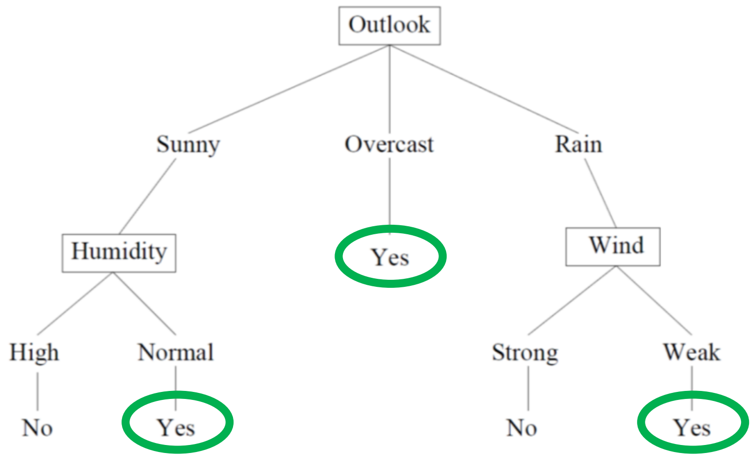
Figure credit: Tom Mitchell, 1997

Decision trees – intuition

- Ask a series of questions based on the values of variables
- Natural way to make decisions
- Effectively divide up the space of all data points into regions to have simple predictions in each region

Notice how we dont need all the inputs to make the decision tree

Reading off the rules

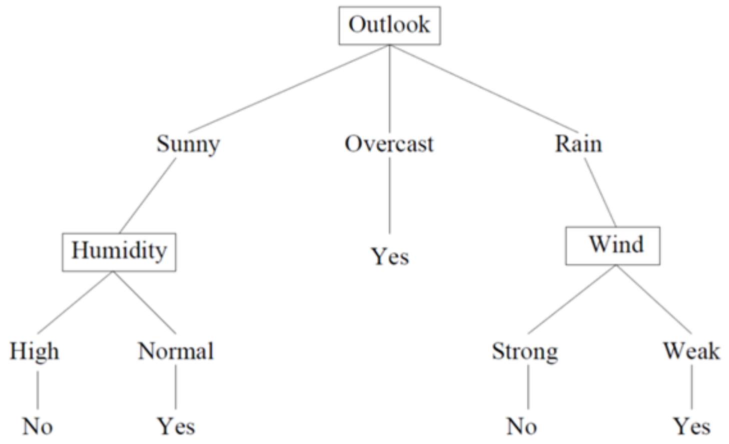


- What rule does this tree encode?
- "We can play tennis if:
    - It is sunny with normal humidity, or
    - It is overcast, or
    - It is raining with weak wind."
- Follow the branches ending with the class we want ("Yes" in this case).
    - *"Or"* across branches ("∨"), *"and"* down branches ("∧")
- $PlayTennis$

$$= (Outlook = sunny \wedge Humidity = normal) \vee (Outlook = overcast) \vee (Outlook = rain \wedge Wind = weak)$$

Classifying with the tree

- Given a new data point…
- Read down the tree
  - Answer questions top-down
- (Sunny, Cool, High, Weak)?
  - No
- (Rain, Mild, Normal, Weak)?
  - Yes
- (Overcast, Hot, Normal, Strong)?
  - Yes



Constructing a tree

- Want to be able to learn a tree
  - From training data
- Involves deciding which questions to ask where
  - What are the best nodes to have higher up?
- Once it has been learned, we can report its accuracy on the testing data

- A common algorithm for learning trees is the ID3 algorithm

The ID3 Algorithm

e.g. first branch for sunny, then all the things for sunny

Main loop:

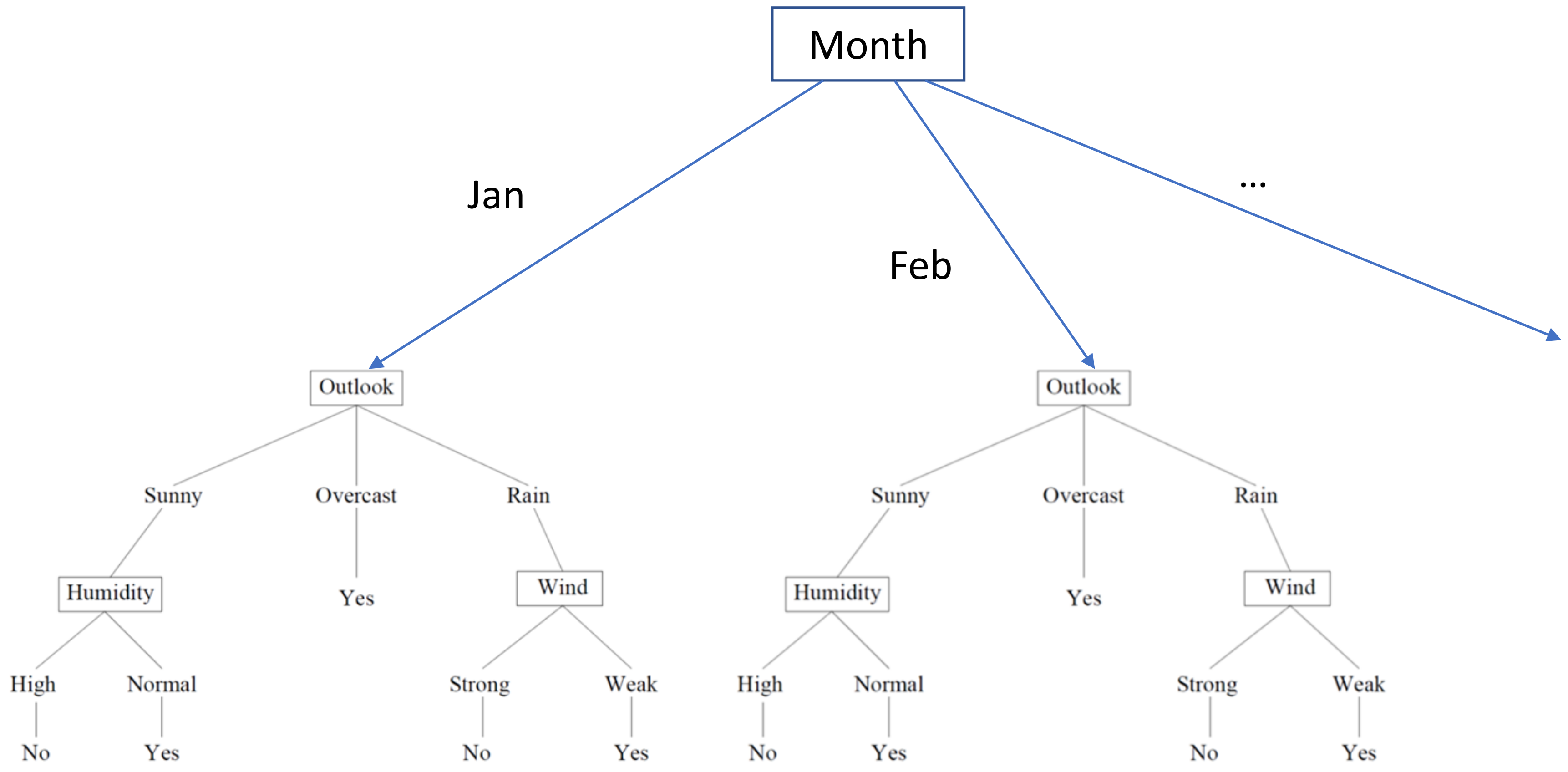
1. F ← “best” feature for next *node*
2. Assign F as decision feature for *node*
3. For each value of F, create new branch from *node*
   - One for each value f that F can take
4. Sort/distribute training examples to leaf nodes
   - Assign all data points that have F=f to branch f
   - Leaf node predicts its most common label
5. If training examples perfectly classified, then STOP, else iterate over new leaf nodes

Choosing the best feature

and you are trying to find Courtney then you get 8billion - 1. But if your first question is male or female? You halve it immediately. Note that are they from South Africa would only work for this example, but is not the best question for every example.

- Why does it make a difference what features we choose?
  - Don’t ask uninformative questions up front: waste of computation, and may lead to having to repeat the same important question later on, on many different branches.
  - E.g. Imagine we first split on another feature “day”, which doesn’t affect the predictions at all. Then we would have 7 copies of the tree.
- How to choose a good feature?
  - We want one that has strong discriminating power, i.e. ends up with branches that are more ordered than before the split

Bad features



Entropy

- Need a measure of how “good” a feature is
- Entropy: measure of randomness or uncertainty in a system
- Let $p = \{p_1, p_2, \dots, p_n\}$ be a set of probabilities, with $\sum_{i=1}^{n} p_i = 1$
- Entropy is $H(p) = -\sum_{i=1}^{n} p_i \log_2 p_i$

tells you how many bits do you need to represent p_i

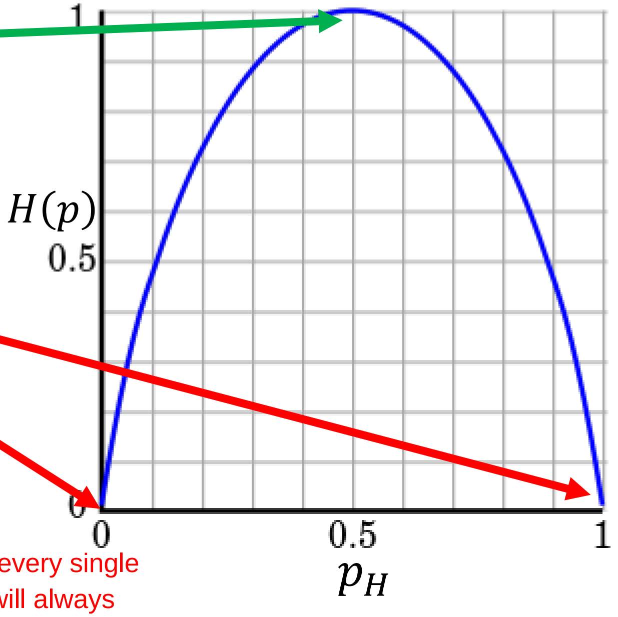
The base 2 in the log is convention, and does NOT depend on the number of outcomes. It is because we measure entropy in “bits”.

Entropy example

entropy encodes how uncertain we are

When its 50/50 we have maximum uncertainty since we are the least likely to know the outcome.
If its 55/45 we have more of a chance of being right by always going with the 55

- Let $p = \{p_H, p_T\}$ be the outcomes of a coin flip
- Entropy is $H(p) = -(p_H \log_2 p_H + p_T \log_2 p_T)$
- Unbiased coin: $p_H = \frac{1}{2}, p_T = \frac{1}{2}$
  - Maximum uncertainty
  - $H(p) = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$
- Maximally biased coin: $p_H = 1, p_T = 0$
  - Minimum uncertainty
  - $H(p) = -1 \log_2 1 - 0 \log_2 0 = 0$



Lower entropy is better! It means we can make more accurate predictions

If p_h is 0 or 1 we know the result every single time, if p_h = 0 then the outcome will always be p_t and vice versa. Entropy is the lowest here

Note: we always take $0 \log_2 0 = 0$
Note: max = 1 only because there are TWO outcomes to a coin flip. Otherwise it would be higher. Why?

Information gain

We want the data point which minimizes our uncertainty (entropy)

- In ID3, we want the feature F which gives the largest reduction in entropy over data D: Gain(D, F)
- Choose feature F with maximum gain
- $Gain(D,F) = H(D) - \frac{1}{|D|}\sum_{f \in values\ of\ F} |D_f| H(D_f)$

Consider gain relative to starting entropy. Note this is the same for every f, so is often left out

Consider all branches: all values of f

Weight by the fraction of the data this constitutes

Entropy of the subset of the data D where F=f

Example

F1	F2	F3	Class
A	S	G	Y
B	S	E	N
A	T	G	N
B	T	G	Y
A	S	G	Y
B	S	E	N

- What is p_Y and p_N?
  - $p_Y = \frac{3}{6}$; $p_N = \frac{3}{6}$
- So, entropy of data H(D):
  - $H(p) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$
  - Also, |D| = 6
- Now compute the gain for each of the three features F1, F2, F3
  - So we can decide which to split on

Example (F1)

- Gain(D, F1)
- f = {A,B}
- For F1 = A (yellow rows):
  - $|D_A| = 3$ no. elements in the row of A
  - $p_Y = \frac{2}{3}\,; p_N = \frac{1}{3}$ prob of yes, no given A
  - $H(D_A) = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.918$
- For F1 = B (purple rows):
  - $|D_B| = 3$ no. elements in rows with B
  - $p_Y = \frac{1}{3}\,; p_N = \frac{2}{3}$ prob of yes,no given B
  - $H(D_B) = -\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} = 0.918$
- $Gain(D, F1) = H(D) - \frac{1}{|D|}\sum_{f\in\{A,B\}}|D_f|H(D_f)$
- $= 1 - \frac{1}{6}\big((3\times 0.918) + (3\times 0.918)\big) = 0.082$

F1	F2	F3	Class
A	S	G	Y
B	S	E	N
A	T	G	N
B	T	G	Y
A	S	G	Y
B	S	E	N

GOOD! You will be right 2/3rds of the time

We have a slight reduction in entropy: splitting on F1 would give us two branches that have slightly less uncertainty: i.e. a 2/3 vs 1/3 split

Example (F2)

F1	F2	F3	Class
A	S	G	Y
B	S	E	N
A	T	G	N
B	T	G	Y
A	S	G	Y
B	S	E	N

- Gain(D, F2)
- f = {S,T}
- For F2 = S (yellow rows):
  - $|D_S| = 4$
  - $p_Y = \frac{1}{2}\,; p_N = \frac{1}{2}$
  - $H(D_S) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$
- For F2 = T (purple rows):
  - $|D_T| = 2$
  - $p_Y = \frac{1}{2}\,; p_N = \frac{1}{2}$
  - $H(D_T) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$
- $Gain(D, F2) = H(D) - \frac{1}{|D|}\sum_{f\in\{S,T\}}|D_f|H(D_f)$
- $= 1 - \frac{1}{6}\big((4\times 1) + (2\times 1)\big) = 0$

We have no reduction in entropy: splitting on F2 would give us two branches with the same uncertainty we have now

Example (F3)

F1	F2	F3	Class
A	S	G	Y
B	S	E	N
A	T	G	N
B	T	G	Y
A	S	G	Y
B	S	E	N

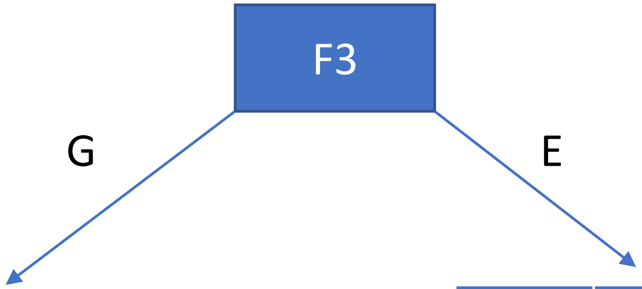
- Gain(D, F3)
- f = {G,E}
- For F3 = G (yellow rows):
  - $|D_G| = 4$
  - $p_Y = \frac{3}{4}\,; p_N = \frac{1}{4}$
  - $H(D_G) = -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.811$
- For F3 = E (purple rows):
  - $|D_E| = 2$
  - $p_Y = \frac{0}{2}\,; p_N = \frac{2}{2}$
  - $H(D_T) = -\frac{0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2} = 0$
- $Gain(D, F3) = H(D) - \frac{1}{|D|}\sum_{f\in\{G,E\}}|D_f|H(D_f)$
- $= 1\ - \frac{1}{6}\big((4 \times 0.811) + (2 \times 0)\big) = 0.459$

We have a large reduction in entropy: splitting on F3 would give us one branch that is all “N”, and another that is 75% “Y”

Example tree

- Gain(D,F1) = 0.082
- Gain(D,F2) = 0
- Gain(D,F3) = 0.459
- Max gain => Split on F3

Now repeat the process on every leaf node that has imperfectly classified training data.
Note the next feature used for splitting may be different on each branch.



Allocate data points to branches where F3 = G/E

F1	F2	F3	Class
A	S	G	Y
A	T	G	N
B	T	G	Y
A	S	G	Y

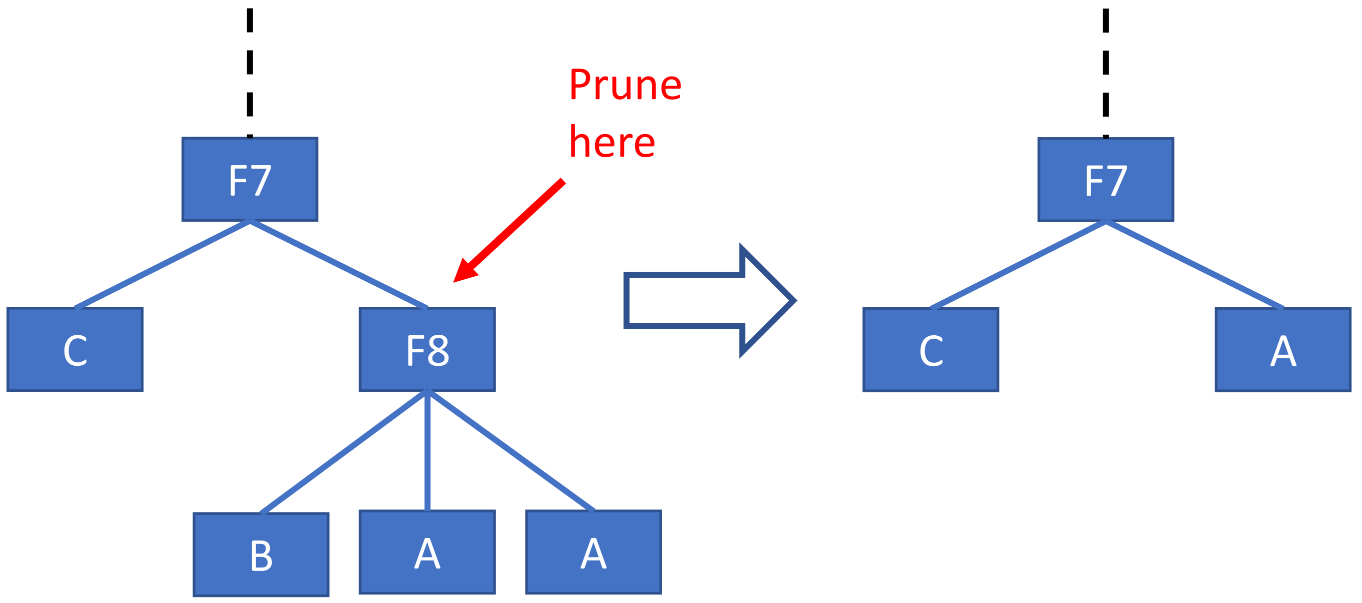
Prediction = "Y" (75% accuracy)

F1	F2	F3	Class
B	S	E	N
B	S	E	N

Prediction = "N"

Avoiding overfitting – pruning

- Overfitting quite likely (particularly when there are only a few data points at a leaf)
- Correct with pruning: replace a subtree with a leaf node with the most common label from the subtree



What to prune?

Remember:
full set of known data = training data + validation data + testing data

- Use ID3 to build tree T using training data
- Compute validation error using validation data:
  (plug each validation point into the tree for a prediction)
- $error_V(T) = \dfrac{number\ of\ misclassifications}{size\ of\ validation\ set}$
- Consider any subtree of T, and prune to give T’
- If $error_V(T') < error_V(T)$:
  - Replace T with T’ and repeat
  - Else: keep T and repeat

- Finally, calculate error on test set for reporting

Continuous variables

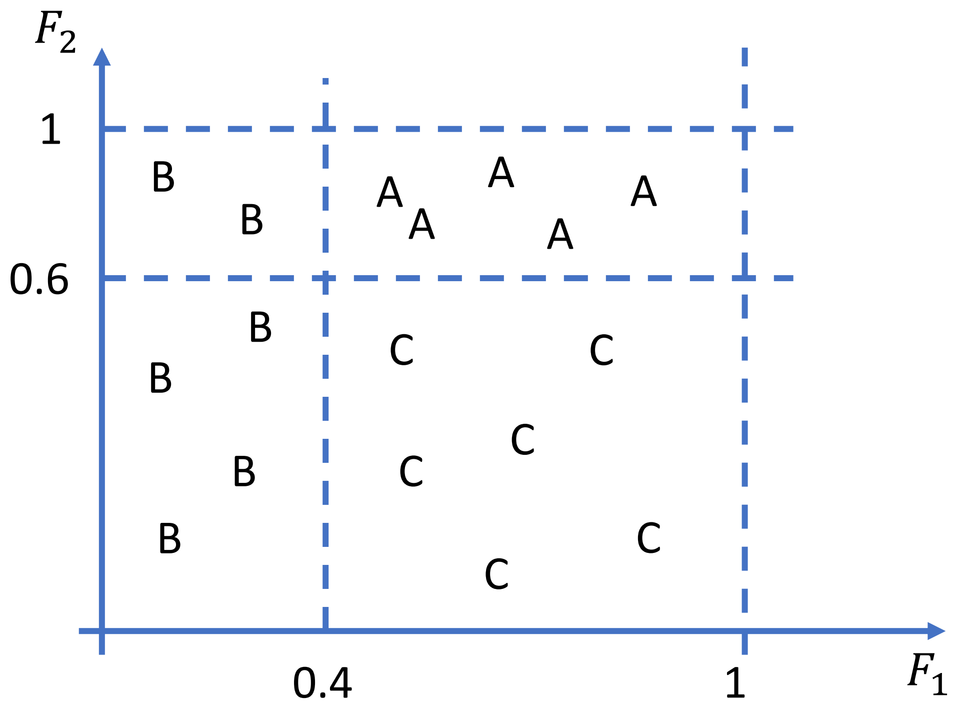
- What if the **attributes** take continuous values?

F_1	F_2	Target
0.83	0.56	A
0.12	0.22	C
0.68	0.19	B
0.79	0.34	A
$\vdots$	$\vdots$	$\vdots$

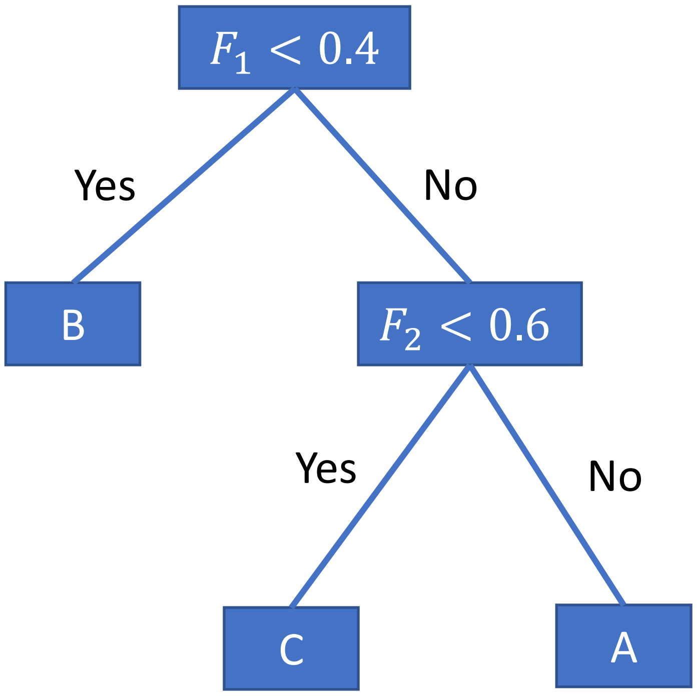
- For each attribute, ask questions like: $F_1 < 0.6$? $F_2 < 0.3$?
- Values 0.6 and 0.3 here are called split-points. How to choose them?
  - Choose possible points over an interval, e.g. 0.1, 0.2, …, 0.9
  - Calculate the Gain of each of these options
    - E.g. Is $F_1 < 0.1$? Is $F_1 < 0.2$? Is $F_1 < 0.3$? …
  - Choose the attribute/split-point with maximum Gain

Continuous variables – example

For two features, we may have:



Giving the tree:



Regression trees

- What if the **target** takes continuous values?
  - This is regression

F_1	F_2	…	F_m	$target$
⋮	⋮	⋮	⋮	2.7
⋮	⋮	⋮	⋮	2.35
⋮	⋮	⋮	⋮	1.98
⋮	⋮	⋮	⋮	-0.33
⋮	⋮	⋮	⋮	-1.05
⋮	⋮	⋮	⋮	0.77
⋮	⋮	⋮	⋮	0.5

- Entropy doesn’t work here!

Sum of squares error

- Instead of entropy, use the sum of squares error: SoSe(D)

- First, calculate the mean of the N targets t_i:
  - $mean = \mu = \frac{1}{N}\sum_{i=1}^{N} t_i$

- Then:
  - $SoSe(D) = \frac{1}{N}\sum_{i=1}^{N}(t_i - \mu)^2$

This error >= 0.
It is minimised when all the targets are at the mean. The error increases as they become spread out.

SoSe example

F_1	F_2	…	F_m	$target$
⋮	⋮	⋮	⋮	2.7
⋮	⋮	⋮	⋮	2.35
⋮	⋮	⋮	⋮	1.98
⋮	⋮	⋮	⋮	-0.33
⋮	⋮	⋮	⋮	-1.05
⋮	⋮	⋮	⋮	0.77
⋮	⋮	⋮	⋮	0.5

- Mean:
- $\mu = \frac{1}{7}(2.7 + 2.35 + 1.98 - 0.33 - 1.05 + 0.77 + 0.5) = 0.989$
- Then, SoSe(D) =
  $$(2.7 - \mu)^2 + (2.35 - \mu)^2 + (1.98 - \mu)^2 + (-0.33 - \mu)^2 + (-1.05 - \mu)^2 + (0.77 - \mu)^2 + (0.5 - \mu)^2 = 11.95$$
- (and then divide by N)

Learning the regression tree

- Same as the ID3 algorithm
- For each attribute F, calculate:
    - $SoSe_F(D) = \sum_{f \in values\ of\ F} |D_f| SoSe(D_f)$
- Choose the F which minimises $SoSe_F(D)$
- Create a branch for each value f of F
- The label given to a leaf node is the **average** of all training values at that node

Summary

- Decision trees as rules
- Classifying with a decision tree
- Building a tree: ID3
- Choosing the best feature: entropy and gain
- Avoiding overfitting: pruning
- Continuous variables: split points
- Regression trees: sum of squares error