* Classification : is taking value and map/output to a discrete label.
eg : True or False
: male or Female

* Classification Learning:

* Instances : INPUT
* Concept (Function) → maps input to outputs : (That defines it (eg: male or female: what male or female is!))
* Target concept : Output, answer, function that
* Hypothesis class : all the functions I am willing to entertain (Classification function)
* Sample : Training set, {Example input, paired with label} {0,1} {F,M}
* Candidate : Target concept
* Testing set : like sample, for prediction.

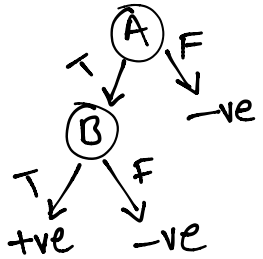
* Decision Trees:

1) Pick the best attribute, the best attribut the eliminate alot of possibilities. Best → splits the data
* best is something that splits in half eg : guess :) living or non-liv
* you asked the next question
3) Follow the answer path
Go to 1 until we get an answer.

* you have to go through all possible questions and answers until the best solution is defined.

* Decision Trees : Expressiness

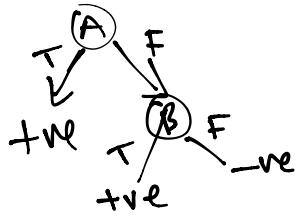
* Boolean
* AND & : A AND B



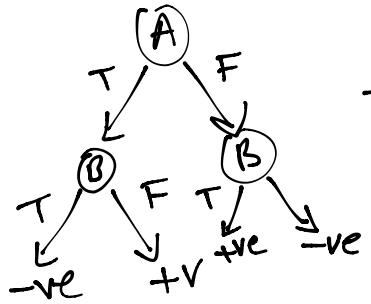
Decision tree for A AND B

* A OR B decision

A' or B are commutative



* A XOR B : exclusive OR, True when A True or B is True. when A True & B True, then False



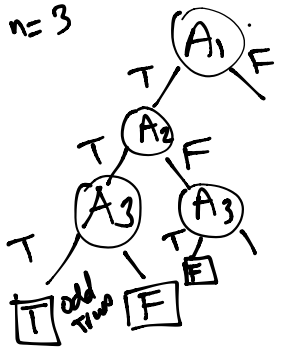
This is a full truth table.

* in case of OR we needed two node and in case of XOR we needed 3 it matter when the size of decision tree grows.

* n-OR : Any, the size of decision tree nodes is linear, n.

* n-XOR : ODD Parity

n=3



$O(2^N)$

Exponential number of nodes, is also called hard.

* therefore in ML we look forward to problems like Any and less like Parity.

* XOR → hard.

* Exactly how expressive is decision tree.
* n attributes (boolean) O(n!) nodes → exp

* how many trees?
* output is boolean

* how many trees

* Answer: # of nodes 2^n

* number of binary combination in the truth table : 2^{2n}
* which grows very fast and is very expressive.

* by 6 there are billions of combinations, hence we need a intuitive combinations.

* ID3:
  * we keep Loops:
    1) Finding the best value attribute
    2) Assign A as decision. : attribute for node
    3) For each value of A
       create a descent of node
    4) Sort Training examples to leaves
    5) If Examples Perfectly classified. STOP

measure of information | Sample label

Information Gain (S, A)

Entropy: measure of ran randomness

Gain(S, A) =

Entropy(S) −

$\sum_v \frac{|S_v|}{|S|}$ Entropy(S_v)

Entropy: $\sum_v$ p(v) log p(v) / value

* ID3 BIAS: Restriction Bias : A ⇐ Preference BIAS
  * Inductive Bias:
    — Good splits at the top
    — Correct Over Inconet
    - Shorter Trees.

* Decision Trees: other consideration:

— Continuous Attributed?
.eg. Age, weight, Distance

* what if we come across a value not in the training data?

* how about Ranges: 

* in truely continuous data there could be infinite number of checks.

* But this is how we evaluate continuous variables/data

* when do we really STOP ⇒ when everything is classified correctly!
  * if there is noise in the data then it may run into an infinite loop.

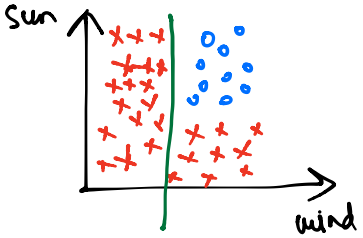
* we have to be careful about Over fitting / Variance.
  * when it violates Occum's Razor.
* how about using cross validation for Ideal curve performance.
* we can go through the entire tree, and the collapse the tree (Pruning) to see how much error is being produced.
  * AFTER PRUNING, we can do an Voting.
* Regression?
  * information Gain may not work well, lets use a different splitting criteria
  * Variance?
  * Output: Avg, Local Linear fit

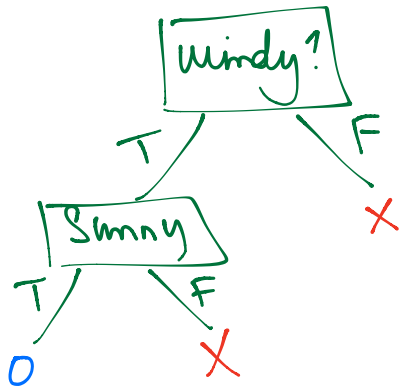
* Recap:

Decision Trees

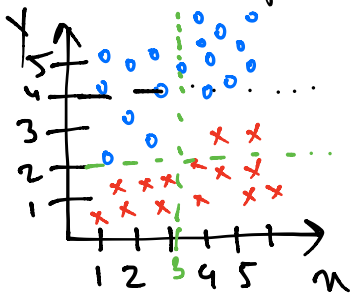
* Representation
* ID3: A TOP Down Learning Algorithm
* Expressiones of DTS
* BIAS of ID3
  * "BEST ATTRIBUTES" like → GAIN (S,A)
  * Dealing with Overfitting.

Decision Tress: we can ask multiple linear questions
* it effectly constructs a linear banier.

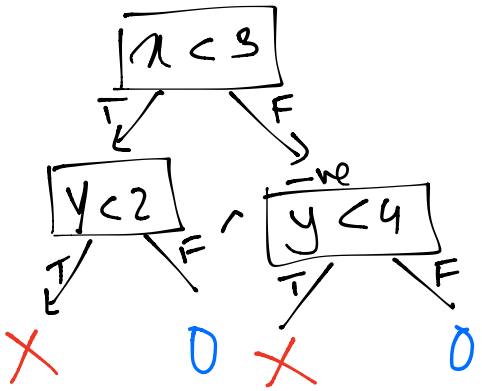




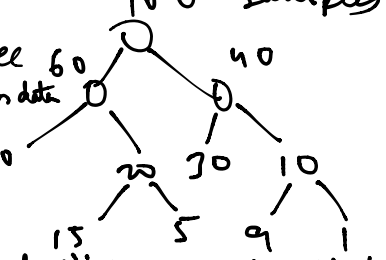
* what is the most useful split, with max Gain — attribute



* x split at 3
* y split at 3 (x<3)



* Decision Tree for: min_samples_split



* Deep in the decision tree complications can occur when data is over splitted. Hence overfitting the data.

* hence we needed to keep the tuning limited by limiting min_sample_split. to higher numbers.

Entropy: the measure which decides where to split the data.

* it is a measure of impurity in a bunch of examples

$\text{Entropy} = \sum_i -P_i \log_2(P_i)$ ← fraction of examples

* it is opposite of purity

* Entropy : same class : entropy = 0.0
  * evenly split between classes → entropy = 1.0 (max value of entropy)

* Entropy : $\text{entropy} = \sum_i (P_i) \log_2 (P_i)$

* eg. slow / fast : SSff : slow nodes : 2 , total nodes : 4

entropy ⇒ P_i is fraction of slow examples ⇒ $P_{i_{slow}} = \frac{2}{4} = 0.5$

$P_{fast} = \frac{2}{4} = 0.5$

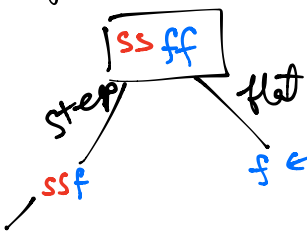
what is the entropy of this node : SSff

$-0.5 \log_2(0.5) - 0.5 \log_2(0.5) = 1$

grade	bumpiness	speed limit	speed
steep	bumpy	yes	slow
steep	smooth	yes	slow
flat	bumpy	no	fast
steep	smooth	no	fast

Gain ⇒ information Gain =

* entropy of parent = 1



$P_{slow} = \frac{2}{3}$

$P_{fast} = 1/3$

$\text{entropy} = \left[-\frac{2}{3} \log_2 \frac{2}{3} - \frac{1}{3} \log_2 1/3 \right]$

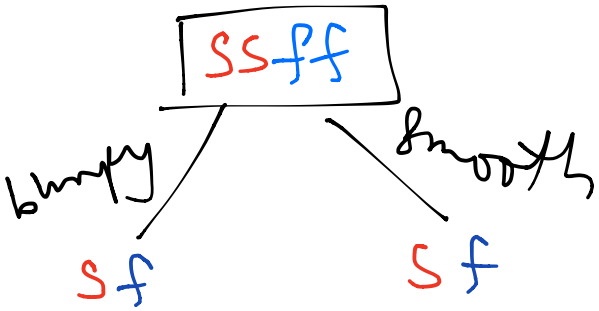
entropy = 0.91829 //

entropy of children) = $\frac{3}{4}(0.91829) + \frac{1}{4}(0) = 0.68872$

* Gain = $1 - 0.68872 = 0.31127$ if we split on grade

entropy on bumpiness :

entropy (bumpiness) :



$P_{slow} = \frac{1}{2}$

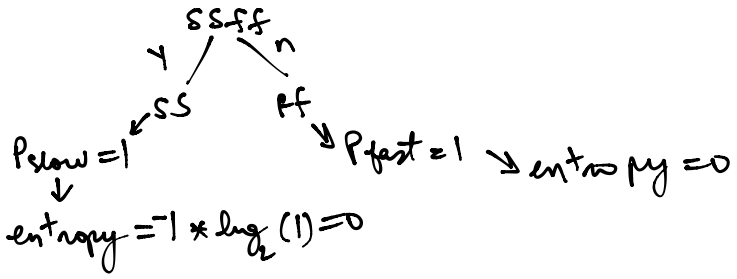
$P_{fast} = \frac{1}{2}$

entropy $= \left[-\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\frac{1}{2}\right)$

$= 1$

entropy of children $= \frac{1}{2}(1) + \frac{1}{2}(1) = 1$

Gain on bumpiness $= 1 - 1 = 0$, we donot want to start spliting starting here.



Gain $= 1 - 0 = 1$ // => max gain

* Bias – Variance trade off – somewhere in the middle, so it is able to generalize the new data and predict.

* DT – stop the growth of the decision trees before it goes too deep by limiting the depth of the tree and min_samples_split