Lecture

Machine Learning
Tom Mitchell
(McGraw-Hill)

Decision Tree

A decision tree is a representation similar to that of a flowchart having three kinds of nodes.

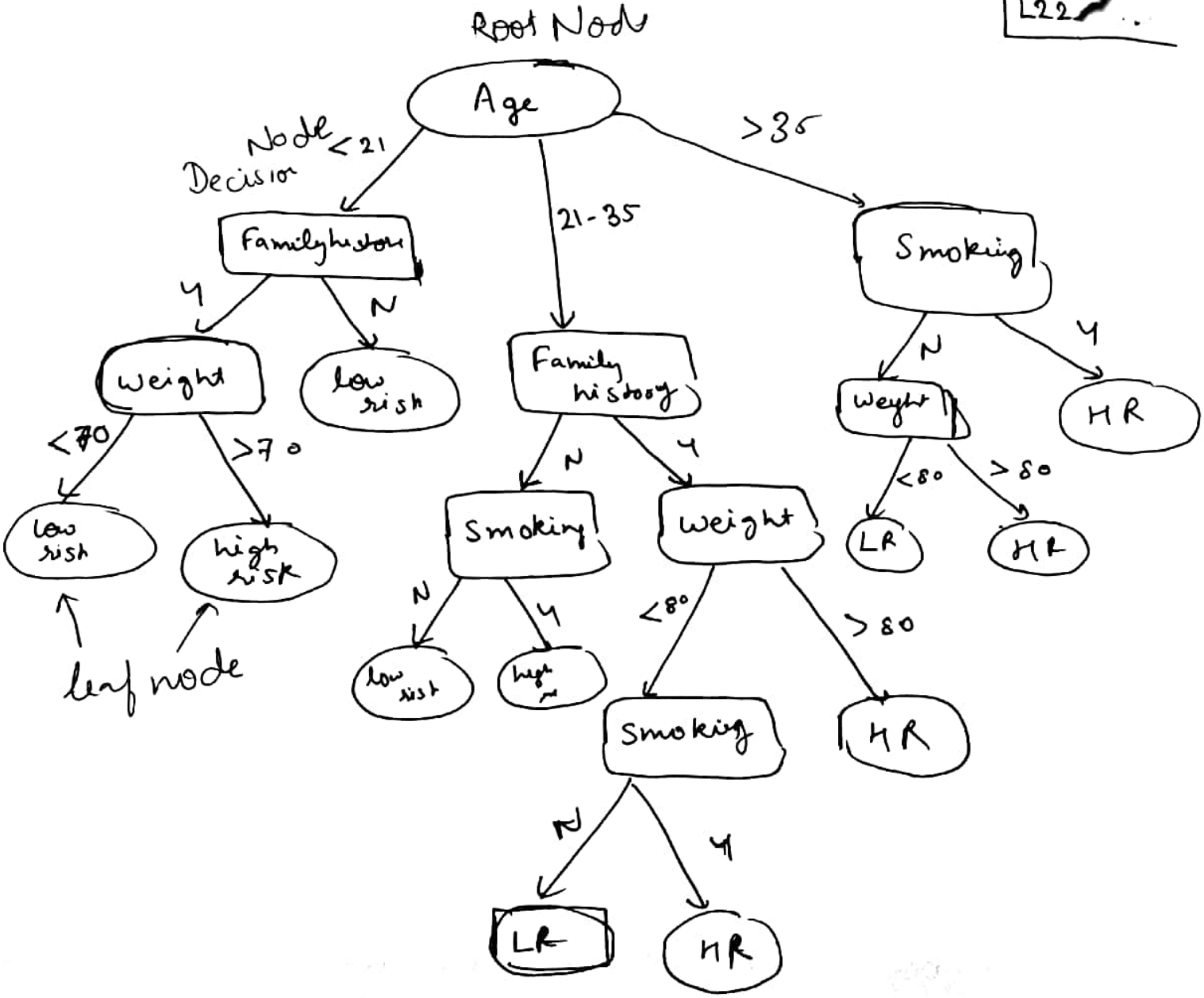
1) Tree nodes
2) Internal nodes
3) LeafNode or Terminal node.

Nodes where no incoming ~~node~~ links are present are tree node.

Those with at least one incoming link internal node

Several incoming link but not outgoing link are leaf nodes.

In decision tree → branches represent the test conducted + labels on leaf node are predicted classes or outcomes.



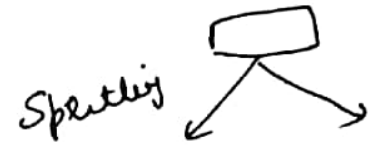
A classification tree - structured

Root Node → node that performs first split

Terminal Node / Leaves → node predicts the outcome.

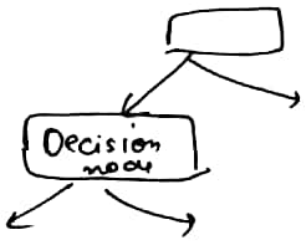
Branches → arrow connect the nodes.

Splitting → Process of dividing node into two or more sub-nodes



L-22-9a

Decision node - sub-node further splits into sub-node



Parent node - node split into sub-node is called the Parent node

⇒ Decision Tree Classifier is capable of both binary classification & multi class classification.

One can provide a training example and generate a decision tree.

→ There can be many decision trees which might fit the training example.

Now which one we should use?

One should try to choose the one with the minimum error.

Further in case there is not much difference in the few models, one can use the Ocam razor.

bias → choose the smaller tree.

Smaller tree

↓

decision trees with smaller number of nodes or trees with smaller depth.

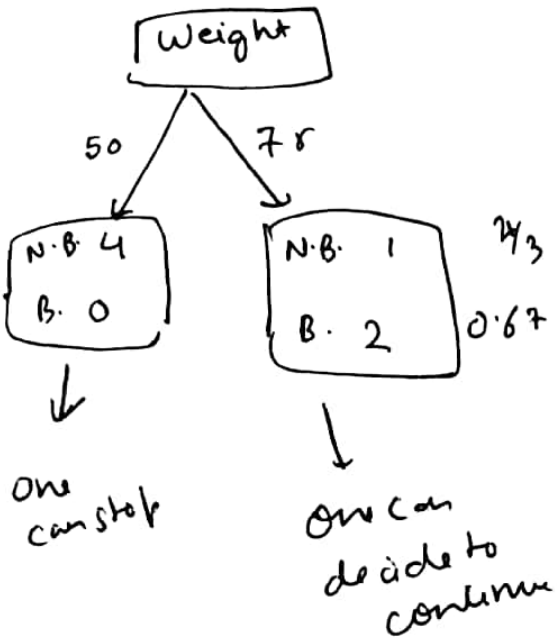
Ideally; given no noise training example, we want to come up with a small tree that fits well with the data.

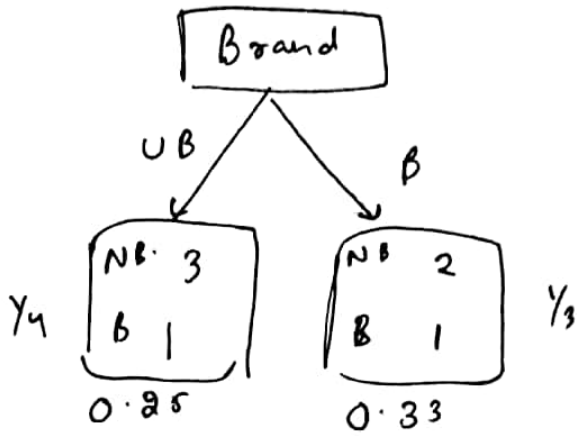
But this is not an easy, it is a computationally hard problem.

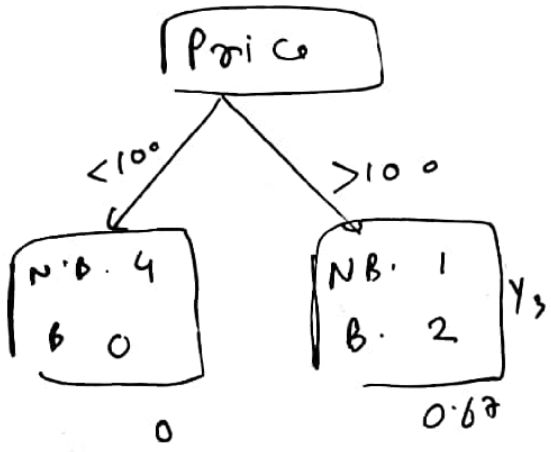
Let see example of purchase in Amazon.

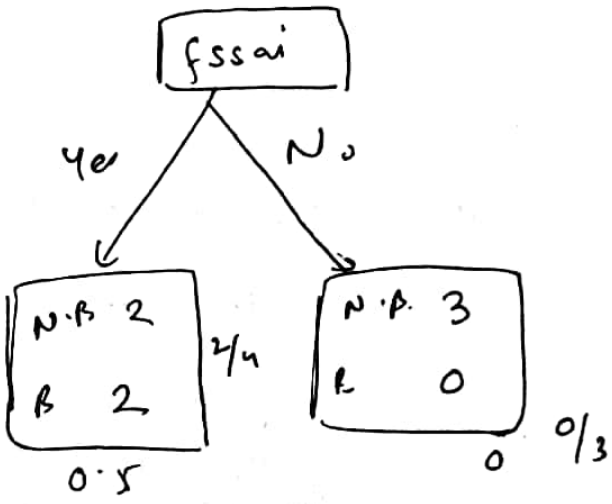
Result	Brand	Price	Weight	fssai
No Buy	Branded	>100	50gm	Yes
Buy	Unbranded	>100	75gm	Yes
N.B.	U.B.	<₹00	50gm	No.
N.B.	B	<100	50gm	Yes
B	B	>100	75gm	Yes
N.B.	B	<100	50gm	No
N.B.	UB	<100	75gm	No
	B	>100	75gm	Yes
N.B.		>100	75gm	Yes

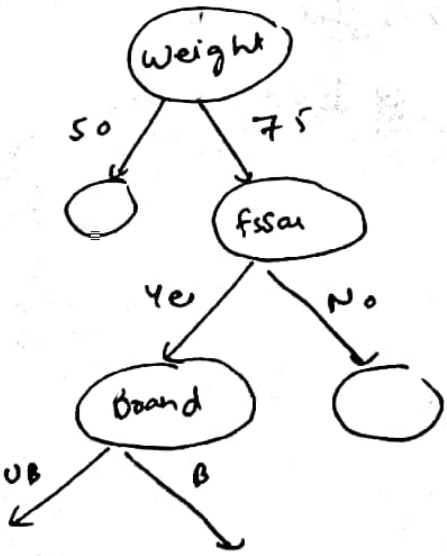
Attributes - Brand, Price, Weight, fssai

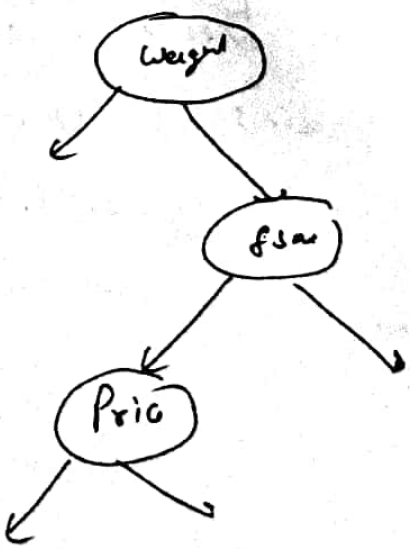


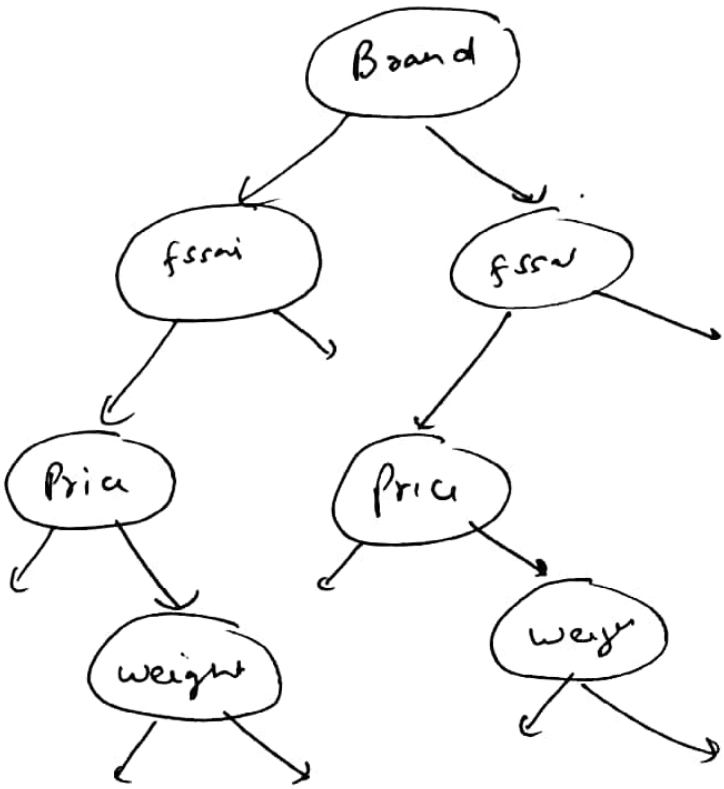












One can build many decision trees. ~~Bad~~ ~~The model~~ But as we see when we use weight or Price as root node, chances are higher for us to have small tree.

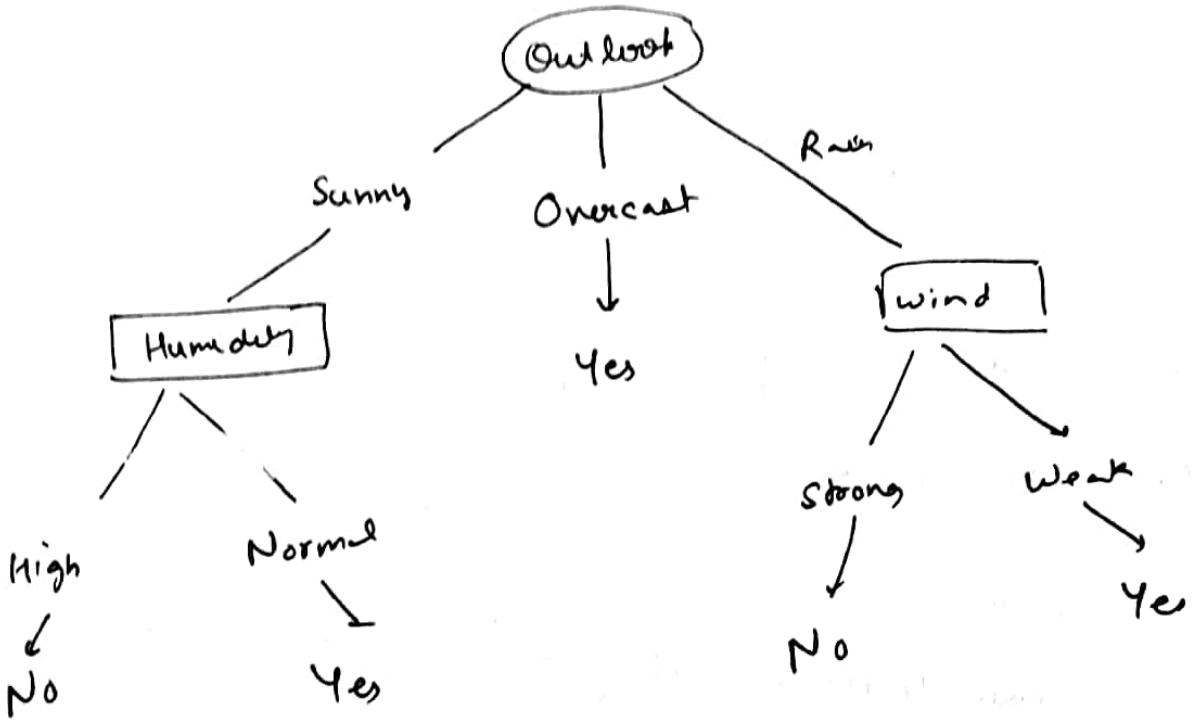
Decision tree is a very flexible function which can represent all the results. ~~If~~ but it can grow to a sufficiently large.

~~One need~~

Data from Tom Mitchel Book

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

Target
↓
Play tennis



One has to find a good decision tree. At any node or point, one has to decide.

→ whether to stop

or → one should continue.

↓

then one should find or choose the most effective attribute or feature

ID3 (Iterative Dichotomiser 3)

algorithm is a simple decision tree algorithm u for classification tasks. It was developed by Ross Quinlan in 1983.

Basic idea of ID3 algorithm is to construct the decision tree by employing a top-down, greedy search through the given sets to test each attribute at every tree node.

→ which node to proceed
↳ "best" decision" attribute for next node.

If all training examples are perfectly classified stop: else iterate.

ID3 algorithm basic
↓
learns decision trees by constructing them

top-down,
start with question " which attribute should be tested at the root of the tree"?

Each instance attribute should be ~~tested~~ evaluated using a statistical test to determine how well it alone classifies the training examples.

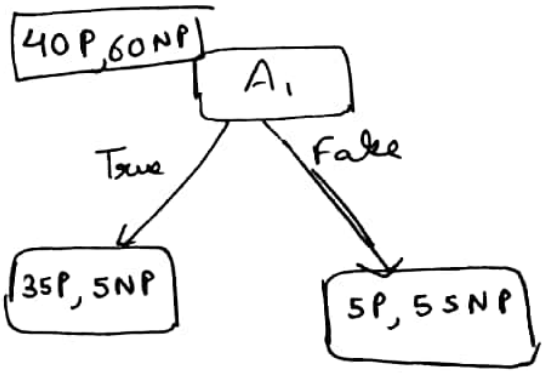
The best attribute is selected and used as the test at the root node of tree.

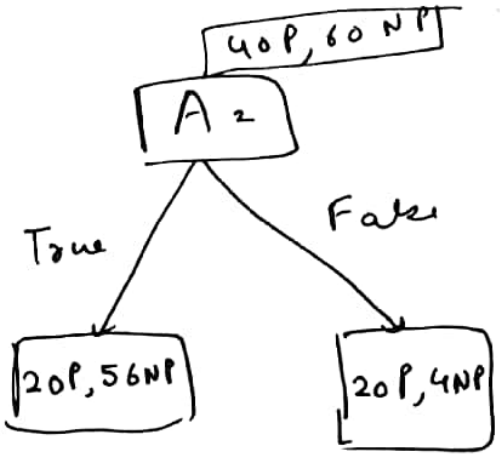
A descendant of root node is then created for each possible value of this attribute and training examples are sorted to the appropriate descendant node.

The entire process is then repeated using the training examples associated with each descendant node to select the best attribute to test at that point in the tree.

This forms a greedy search for an acceptable decision tree, in which algorithm never backtracks to reconsider earlier choices

Let we have many attributes $A_1, A_2, \ldots A_n$
Let's say only 2.
based on it if True or False
on purchase (P)
or No purchase (NP)





100 training sample (40 purchase + 60 No purchase)

Based on the above information one has to decide which one is to be used first.

One can utilize multiple methods.

Two famous methods are

1) Entropy 2) Gini

2) ~~Information Gain~~

Entropy → A measure of uncertainty, impurity + information content.

In information theory, entropy characterizes the (im)purity of an arbitrary collection of examples.

Given a collection S, containing positive + negative examples of some target concept.

Entropy of S relative to this boolean classification is

→ $Entropy(S) = -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$

$p_{\oplus}$ is proportion of +ve examples in S

$p_{\ominus}$ is proportion of -ve example in S.

In calculation, we define $0 \log 0$ to be 0.

If $p_{\oplus} = 1$, $p_{\ominus} = 0$

$Entropy(S) = 0$

when $p_{\ominus} = 1$, $p_{\oplus} = 0$

$Entropy(S) = 0$.

when $p_{\ominus} = \frac{1}{2}$, $p_{\oplus} = \frac{1}{2}$

$Entropy(S) = 1$ (highest)

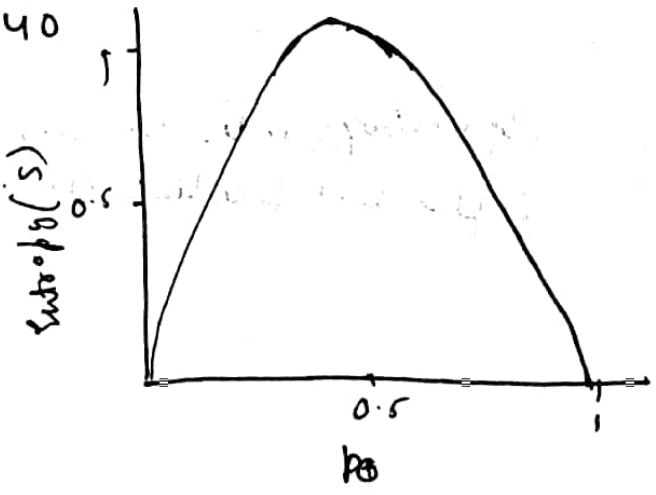
S is collection of 14 example of some boolean.

9⊕ 5⊖.

$Entropy(9+, 5-) = (-9/14) \log_2 (9/14) - (5/14) \log_2 (5/14)$

$= 0.940$

$Entropy(14+, 0-) = 1$



One interpretation of entropy from information theory is that it specifies the minimum no of bits of information needed to encode the classification of an arbitrary member of S.

If $p_{\oplus}$ is 1, no message need to be send

If $p_{\oplus}$ is 0.5; one bit required to indicate whether drawn example is ⊕ or ⊖.

If $p_{\oplus}$ is 0.8; collection of messages can be encoded using on average less than 1 bit per message by assuming shorter code to collection of ⊕ & longer codes to less likely ⊖.

shorter code for more frequently occurry
longer code for less frequently occurry

More efficient encoding

↓

Huffman coding is used in data compression & information theory

High entropy indicates more secure data as it implies less predictability.

We discussed till now the entropy in the special case where target classification is boolean If target attribute can take on c different values. then the entropy of S is defined as

$$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

$p_i \rightarrow$ proportion of S belonging to class i.

logarithm is still base 2 ∵ entropy is a measure of the expected coding length measured in bits.

If target attribute can take on c possible values, the entropy can be as large as $\log_2 c$.

Entropy is maximum, when one has no knowledge of the system.

Information Gain

Given entropy as a measure of impurity in a collection of training examples, we can now define a measure of effectiveness of an attribute in classifying the training data.

Information Gain → simply the expected reduction in entropy caused by partitioning the examples according to the particular attribute

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

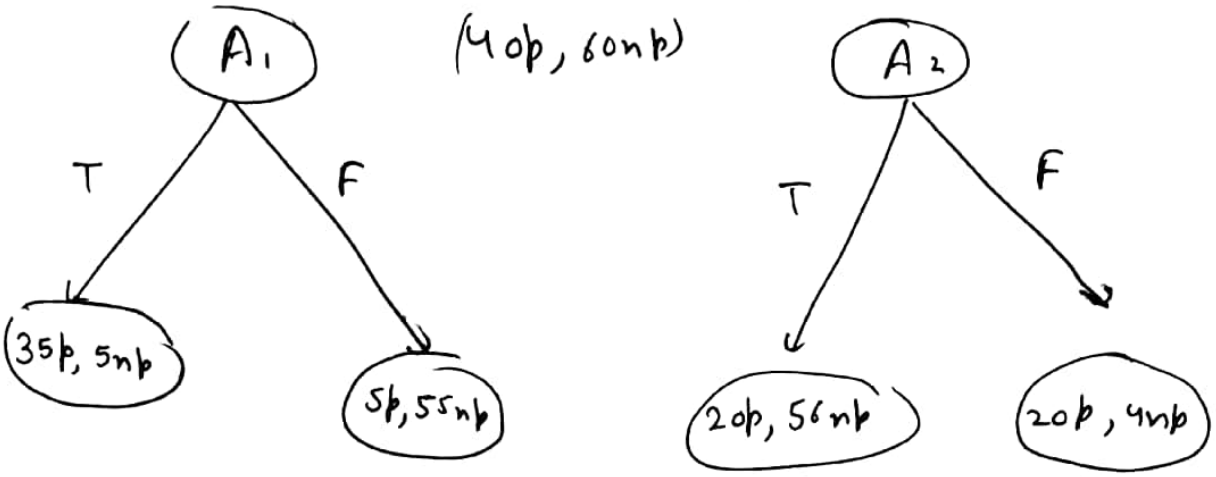
Values(A) is set of all possible values of for attribute A, and S_v is the subset of S for which attribute A has value v

i.e. $S_v = \{ s \in S | A(s) = v \}$

Expected entropy is simply the sum of entropies of each subset S_v, weighted by fraction of examples $\frac{|S_v|}{|S|}$ that belong to S_v.

a)

Gain (S, A) is the expected reduction in entropy caused by knowing the value of attribute A



Entropy (35p, 5np) = 0.54
Entropy (5p, 55np) = 0.41

Entropy (20p, 56np) = 0.83
Entropy (20p, 4np) = 0.65

$$\text{Entropy}(40p, 60np) = 0.97$$

$$\text{Gain}(S, A_1) = \text{Entropy}(40p, 60np) - \frac{40}{100} \times \text{Entropy}(35p, 5np) - \frac{60}{100} \times \text{Entropy}(5p, 55np)$$

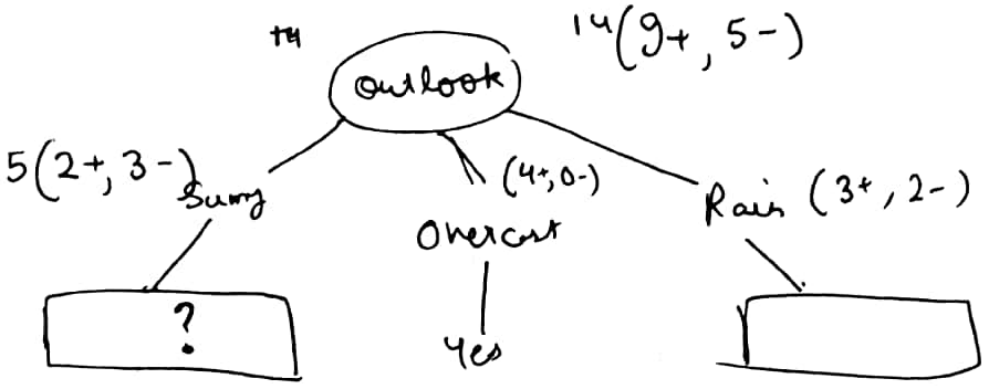
$$= 0.97 - 0.4 \times 0.54 - 0.6 \times 0.41$$

$$= 0.508$$

$$\text{Gain}(S, A_2) = \text{Entropy}(40p, 60np) - \frac{76}{100} \times \text{Entropy}(20p, 56np) - \frac{24}{100} \times \text{Entropy}(20p, 4np)$$

$$= 0.183$$

Based on this we should use A_1 as the Gain is large



→ Temperature
→ Humidity
→ Wind

Gain (S, Outlook) = 0.246

$$S(9,5-) - \frac{5}{14} S(2,3) - \frac{4}{14} S(4,0) - \frac{5}{14} S(3,2)$$

Gain (S, Humidity) = 0.151

Gain (S, Wind) = 0.048

Gain (S, Temp.) = 0.029

According to the information gain measure, the Outlook attribute provide the best prediction of the target attribute

∴ Outlook is selected as the Root node.

:|

Gain (S_{sunny}, Humidity) = 0.970

Gain (S_{sun}, Temp) = 0.570

Gain (S_{sunny}, Wind) = 0.019.

Therefore. After sunny one uses Humidity as decision node.

There are other measures one can utilize to decide the attribute for decision tree.

GINI index :-

It measure the inequality of distribution. It is a measurement of the purity of nodes & is used for all dependent variables.

$$GINI_{node} = 1 - \sum_{c \in classes} |p(c)|^2$$

$p(c)$ → probability of class c

↓

fraction of examples belonging to the class.

Gini Split

↓ often referred to Gini gain or Gini reduction.

↓ measures the reduction in impurity achieved by splitting a node into child nodes. based on a specific attribute.

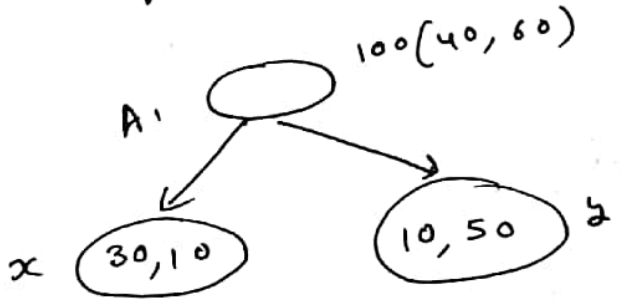
$$\text{Gini Split} = \text{Gini(parent node)} - \sum \frac{|S_v|}{|S|} \text{Gini}(S_v)$$

Gini(parent node) → Gini index of parent node

$|S_v|$ is the number of instances in child node v

$|S|$ is number of instances in parent node

Gini (S_v) → Gini index of child node v



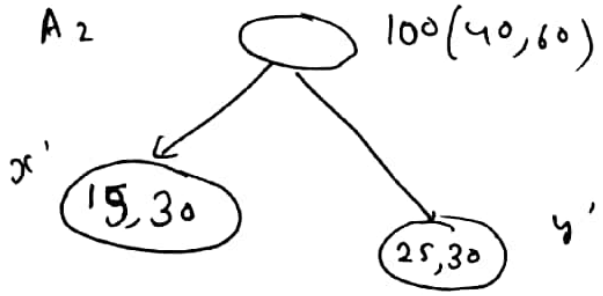
$$\text{Gini(node)} = 1 - (0.4^2 + 0.6^2) = 0.48$$

$$\text{Gini}(x) = 1 - \left(\left(\frac{3}{4}\right)^2 + \left(\frac{1}{4}\right)^2\right) = 0.375$$

$$\text{Gini}(y) = 1 - \left(\left(\frac{1}{6}\right)^2 + \left(\frac{5}{6}\right)^2\right) = 0.278$$

$$\text{Gini split} = 0.48 - \frac{4}{10} \times 0.375 - \frac{6}{10} \times 0.278$$

$$= 0.163$$



Gini (node) = 0.48

Gini (x') = $1 - \left(0.33^2 + 0.66^2\right) = 0.444$

Gini (y') = $1 - (0.45^2 + 0.54^2) = 0.4959$

Gini split = $0.48 - 0.45 \times 0.444 - 0.55 \times 0.4959$

= 0.007

Higher Gini split better.

In some cases people use

Gini split = $\sum \frac{|S_v|}{|S|}$ Gini (S_v)

in that case lower Gini split is better.

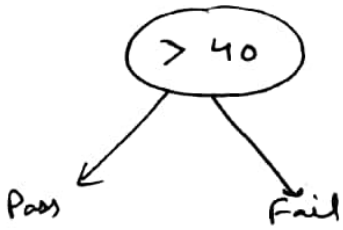
They are almost same.

Currently we are saying that the attribute is binary or discrete.

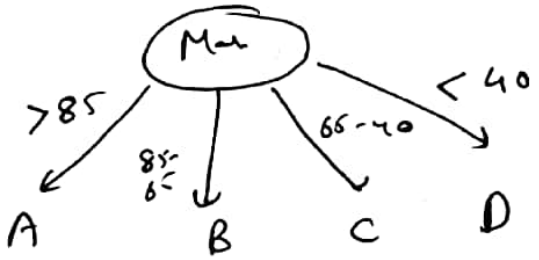
In real life it can be real & continuous number.

e.g.

Marks



Binary split



Multi way split

Let say one want to divide into two ranges of continuous attribute

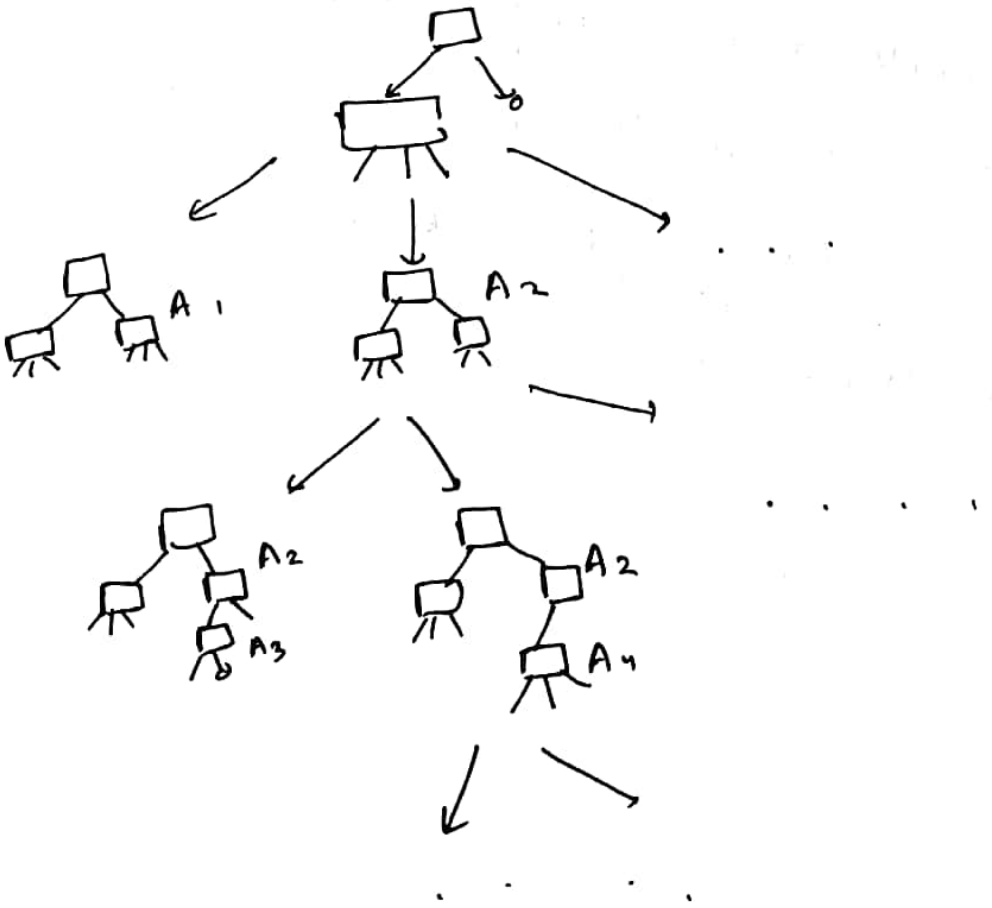
$$A_c = \begin{cases} \text{true} & \text{if } A_c < c \\ \text{false} & \text{otherwise} \end{cases}$$

How one can choose c value.

One can here also uses the c value which give the highest level of gain in the information.

ID3 → searching a space of hypotheses for one that fits the training examples

ID3 performs simple-to-complex, hill climbing search through the hypothesis space, beginning with empty tree, then considering progressively more elaborate hypothesis in search of a decision tree that correctly classifies the training data.



L22 - a

ID3 in its pure form performs no backtracking in its search.

↓

chance of converging to local optimal instead of global.

let say we have an algorithm which give many trees + one select the smallest consistent tree. This will have a bias "shortest trees are prefered over longer".

ID3 does same by using its greedy heuristic search to ~~attempt~~ find shortest tree without conducting the entire search through the hypothesis space.

It exhibits a more complex bias.

It not only select smallest but also favor trees that place attributes with high information gain closest to the root.